In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
file = 'diabetes_prediction_dataset.csv'
db = pd.read_csv(file)
```

Data Preprocessing

In [ ]:
```
db.head(20)
```

Out[ ]:

	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
5	Female	20.0	0	0	never	27.32	6.6	85	0
6	Female	44.0	0	0	never	19.31	6.5	200	1
7	Female	79.0	0	0	No Info	23.86	5.7	85	0
8	Male	42.0	0	0	never	33.64	4.8	145	0
9	Female	32.0	0	0	never	27.32	5.0	100	0
10	Female	53.0	0	0	never	27.32	6.1	85	0
11	Female	54.0	0	0	former	54.70	6.0	100	0
12	Female	78.0	0	0	former	36.05	5.0	130	0
13	Female	67.0	0	0	never	25.69	5.8	200	0
14	Female	76.0	0	0	No Info	27.32	5.0	160	0
15	Male	78.0	0	0	No Info	27.32	6.6	126	0
16	Male	15.0	0	0	never	30.36	6.1	200	0
17	Female	42.0	0	0	never	24.48	5.7	158	0
18	Female	42.0	0	0	No Info	27.32	5.7	80	0
19	Male	37.0	0	0	ever	25.72	3.5	159	0

In [ ]:
```
db.shape
```

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

In [ ]:
```
db.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
age	100000.0	41.885856	22.516840	0.08	24.00	43.00	60.00	80.00
hypertension	100000.0	0.074850	0.263150	0.00	0.00	0.00	0.00	1.00
heart_disease	100000.0	0.039420	0.194593	0.00	0.00	0.00	0.00	1.00
bmi	100000.0	27.320767	6.636783	10.01	23.63	27.32	29.58	95.69
HbA1c_level	100000.0	5.527507	1.070672	3.50	4.80	5.80	6.20	9.00
blood_glucose_level	100000.0	138.058060	40.708136	80.00	100.00	140.00	159.00	300.00
diabetes	100000.0	0.085000	0.278883	0.00	0.00	0.00	0.00	1.00

In [ ]:
```
db.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
```

In [ ]:
```
db.age = db.age.astype(int)
```

In [ ]:
```
db.groupby('diabetes').count()
```

Out[ ]:

diabetes	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level
0	91500	91500	91500	91500	91500	91500	91500	91500
1	8500	8500	8500	8500	8500	8500	8500	8500

In [ ]:
```
pd.set_option('display.max_rows', 200)
```

In [ ]:
```
db_no = db.query("diabetes == 0")
db_no = db_no.iloc[0:8500]
db_no.shape
```

Out[ ]: (8500, 9)

In [ ]:
```
db_yes = db.query("diabetes == 1")
db_yes.shape
```

Out[ ]: (8500, 9)

In [ ]:
```
db_predict = pd.concat([db_yes, db_no])
db_predict = db_predict.reset_index(drop=True)
db.shape
```

Out[ ]: (100000, 9)

In [ ]:
```
db_predict.head()
```

Out[ ]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	44	0	0	never	19.31	6.5	200	1
1	Male	67	0	1	not current	27.32	6.5	200	1
2	Male	50	1	0	current	27.32	5.7	260	1
3	Male	73	0	0	former	25.91	9.0	160	1
4	Female	53	0	0	former	27.32	7.0	159	1

In [ ]:
```
db_predict['smoking_history'].value_counts()
```

Out[ ]:
```
never          6309
No Info        4632
former         2341
current        1735
not current    1192
ever            791
Name: smoking_history, dtype: int64
```

In [ ]:
```
gender = {'Male':1, 'Female':0}
smoking_history = {'never':1, 'No Info':2, 'former':3, 'current':4, 'not current':5, 'ever':6}

db_predict['gender'] = db_predict.loc[:,'gender'].map(gender)
db_predict['smoking_history'] = db_predict.loc[:,'smoking_history'].map(smoking_history)
```

In [ ]:
```
db_predict
```

Out[ ]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	0	44	0	0	1	19.31	6.5	200	1
1	1	67	0	1	5	27.32	6.5	200	1
2	1	50	1	0	4	27.32	5.7	260	1
3	1	73	0	0	3	25.91	9.0	160	1
4	0	53	0	0	3	27.32	7.0	159	1
...	...	...	...	...	...	...	...	...	...
16995	1	62	0	0	2	27.32	4.0	145	0
16996	1	45	0	0	2	27.32	6.1	90	0
16997	0	46	0	0	1	27.32	4.0	159	0
16998	0	16	0	0	2	22.95	5.8	155	0
16999	0	63	0	0	2	34.40	6.0	100	0

17000 rows × 9 columns

In [ ]:
```
X = db_predict.iloc[:,:-1].values
y = db_predict.iloc[:,-1].values
```

Machine Learning Classifiers

In [ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [ ]:
```
# Split the dateset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=12)
```

In [ ]:
```
# Scale the dataset
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Decision Tree Classifier

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [ ]:
```
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_pred = dtc.predict(X_test)
```

In [ ]:
```
score = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
```

In [ ]:
```
print('Confusion Matrix')
print('    0 \t  1')
print(f'0 {cm[0]}')
print(f'1 {cm[1]}')

print(f'Accuracy Score: {np.floor(score * 100)}%')
```

```
Confusion Matrix
    0     1
0 [1473  220]
1 [ 201 1506]
Accuracy Score: 87.0%
```

Random Forest Classifier

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [ ]:
```
rdc = RandomForestClassifier(n_estimators = 100, random_state = 12)
rdc.fit(X_train, y_train)
y_pred = rdc.predict(X_test)
```

In [ ]:
```
score = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
```

In [ ]:
```
print('Confusion Matrix')
print('    0 \t  1')
print(f'0 {cm[0]}')
print(f'1 {cm[1]}')

print(f'Accuracy Score: {np.floor(score * 100)}%')
```

```
Confusion Matrix
    0     1
0 [1523  170]
1 [ 140 1567]
Accuracy Score: 90.0%
```

SVM Classifier

In [ ]:
```
from sklearn.svm import SVC
```

In [ ]:
```
kernels = ['sigmoid', 'linear', 'poly', 'rbf']

for kernel in kernels:
    svm = SVC(kernel=kernel)
    svm.fit(X_train, y_train)
    y_pred = svm.predict(X_test)

    score = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)

    print(kernel.title())
    print('Confusion Matrix')
```

```
    print('    0 \t  1')
    print(f'0 {cm[0]}')
    print(f'1 {cm[1]}')

    print(f'Accuracy Score: {np.floor(score * 100)}%')
    print('')
```

```
Sigmoid
Confusion Matrix
    0     1
0 [1417  276]
1 [ 287 1420]
Accuracy Score: 83.0%

Linear
Confusion Matrix
    0     1
0 [1490  203]
1 [ 200 1507]
Accuracy Score: 88.0%

Poly
Confusion Matrix
    0     1
0 [1508  185]
1 [ 171 1536]
Accuracy Score: 89.0%

Rbf
Confusion Matrix
    0     1
0 [1486  207]
1 [ 152 1555]
Accuracy Score: 89.0%
```

Gradient Boosting Classifier

In [ ]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [ ]:
```
gbc = GradientBoostingClassifier()
gbc.fit(X_train, y_train)
y_pred = gbc.predict(X_test)
```

In [ ]:
```
score = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
```

In [ ]:
```
print('Confusion Matrix')
print('    0 \t  1')
print(f'0 {cm[0]}')
print(f'1 {cm[1]}')

print(f'Accuracy Score: {np.floor(score * 100)}%')
```

```
Confusion Matrix
    0     1
0 [1524  169]
1 [ 120 1587]
Accuracy Score: 91.0%
```

Naive Bayes

In [ ]:
```
from sklearn.naive_bayes import GaussianNB
```

In [ ]:
```
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred = nb_classifier.predict(X_test)
```

In [ ]:
```
score = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
```

In [ ]:
```
print('Confusion Matrix')
print('    0 \t  1')
print(f'0 {cm[0]}')
print(f'1 {cm[1]}')

print(f'Accuracy Score: {np.floor(score * 100)}%')
```

```
Confusion Matrix
    0     1
0 [1510  183]
1 [ 390 1317]
Accuracy Score: 83.0%
```

K-Nearest Neighbors (KNN) Classifier

In [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [ ]:
```
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(X_train, y_train)
y_pred = knn_classifier.predict(X_test)
```

In [ ]:
```
score = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
```

In [ ]:
```
print('Confusion Matrix')
print('    0 \t  1')
print(f'0 {cm[0]}')
print(f'1 {cm[1]}')

print(f'Accuracy Score: {np.floor(score * 100)}%')
```

```
Confusion Matrix
    0     1
0 [1481  212]
1 [ 171 1536]
Accuracy Score: 88.0%
```