```python
import pandas
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score
from sklearn.utils import resample
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.naive_bayes import GaussianNB
from sklearn import svm

df = pd.read_csv(r"C:\Users\nimes\OneDrive\Documents\datasets\
diabetes_dataset.csv")
df.describe()
```

```
               Age  Pregnancies          BMI      Glucose
BloodPressure  \
count  9538.000000  9538.000000  9538.000000  9538.000000
9538.000000
mean     53.577584     7.986161    27.052364   106.104183
84.475781
std      20.764651     4.933469     5.927955    21.918590
14.123480
min      18.000000     0.000000    15.000000    50.000000
60.000000
25%      36.000000     4.000000    22.870000    91.000000
74.000000
50%      53.000000     8.000000    27.050000   106.000000
84.000000
75%      72.000000    12.000000    31.180000   121.000000
94.000000
max      89.000000    16.000000    49.660000   207.200000
138.000000

             HbA1c          LDL          HDL  Triglycerides  \
count  9538.000000  9538.000000  9538.000000    9538.000000
mean      4.650661   100.133456    49.953418     151.147746
std       0.476395    29.911910    15.242194      48.951627
min       4.000000   -12.000000    -9.200000      50.000000
25%       4.300000    80.100000    39.700000     117.200000
50%       4.600000    99.900000    50.200000     150.550000
75%       5.000000   120.200000    60.200000     185.100000
max       6.900000   202.200000   107.800000     345.800000

       WaistCircumference  HipCircumference          WHR
FamilyHistory  \
count         9538.000000       9538.000000  9538.000000
9538.000000
mean            93.951678        103.060621     0.917400
```

```
0.302474
std           15.594468         13.438827     0.140828
0.459354
min           40.300000         54.800000     0.420000
0.000000
25%           83.400000         94.000000     0.820000
0.000000
50%           93.800000        103.200000     0.910000
0.000000
75%          104.600000        112.100000     1.010000
1.000000
max          163.000000        156.600000     1.490000
1.000000

          DietType  Hypertension  MedicationUse      Outcome
count  9538.000000   9538.000000    9538.000000  9538.000000
mean      0.486161      0.001048       0.405012     0.344097
std       0.661139      0.032364       0.490920     0.475098
min       0.000000      0.000000       0.000000     0.000000
25%       0.000000      0.000000       0.000000     0.000000
50%       0.000000      0.000000       0.000000     0.000000
75%       1.000000      0.000000       1.000000     1.000000
max       2.000000      1.000000       1.000000     1.000000
```

```
target_column = "Outcome"
X = df.drop(columns = target_column)
y = df[target_column]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
y_pred = log_reg.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred))
```

```
Logistic Regression Accuracy: 0.9722222222222222

C:\Users\nimes\PycharmProjects\college\.venv\Lib\site-packages\
sklearn\linear_model\_logistic.py:465: ConvergenceWarning: lbfgs
failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
rf_reg = RandomForestRegressor()
rf_reg.fit(X_train, y_train)
y_pred = rf_reg.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred))
```

```
Random Forest Accuracy: 1.0
```

```
dt_reg = DecisionTreeRegressor()
dt_reg.fit(X_train, y_train)
y_pred = dt_reg.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred))
```

```
Decision Tree Accuracy: 1.0
```

```
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred = nb.predict(X_test)
print("Naïve Bayes Accuracy:", accuracy_score(y_test, y_pred))
```

```
Naïve Bayes Accuracy: 0.960167714884696
```

```
svm_model = svm.SVC(kernel='rbf', C=10, gamma=0.1)
svm_model.fit(X_train, y_train)
y_pred = svm_model.predict(X_test)
print("SVM Accuracy:", f1_score(y_test, y_pred))
```

```
SVM Accuracy: 0.0
```

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
print(pd.Series(rf.feature_importances_,
index=df.drop(columns=["Outcome"]).columns).sort_values(ascending=False))
```

```
FamilyHistory         0.824517
Glucose               0.112114
HbA1c                 0.021671
BMI                   0.007321
WaistCircumference    0.004873
BloodPressure         0.004463
HipCircumference      0.004433
Age                   0.003975
Triglycerides         0.003582
LDL                   0.003408
WHR                   0.003220
HDL                   0.003127
Pregnancies           0.001815
MedicationUse         0.000957
DietType              0.000493
```

```
Hypertension          0.000028
dtype: float64
```

The observations from this notebook shows that only logistic regression and naive bayes can provide a optimal solution both ranging inbetween 96 to 98 % accuracy, whereas random forest, decision tree and SVM provides the most unreliable outcome.