```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,confusion_matrix
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
```

```
column_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction'
df= pd.read_csv('/content/diabetes_prediction_dataset.csv', names=column_names)
```

```
<ipython-input-35-46932afbe04d>:2: DtypeWarning: Columns (1,2,3,5,6,7,8) have mixed types. Specify dtype option on import or set low
  df= pd.read_csv('/content/diabetes_prediction_dataset.csv', names=column_names)
```

```
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcom
0	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabete
1	Female	80.0	0	1	never	25.19	6.6	140	
2	Female	54.0	0	0	No Info	27.32	6.6	80	
3	Male	28.0	0	0	never	27.32	5.7	158	
4	Female	36.0	0	0	current	23.45	5.0	155	
...	...	...	...	...	...	...	...	...	
99996	Female	80.0	0	0	No Info	27.32	6.2	90	
99997	Female	2.0	0	0	No Info	17.37	6.5	100	
99998	Male	66.0	0	0	former	27.83	5.7	155	
99999	Female	24.0	0	0	never	35.42	4.0	100	
100000	Female	57.0	0	0	current	22.43	6.6	90	

100001 rows × 9 columns

```
df.dtypes
```

```
Pregnancies                 object
Glucose                     object
BloodPressure               object
SkinThickness               object
Insulin                     object
BMI                         object
DiabetesPedigreeFunction    object
Age                         object
Outcome                     object
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
lst =['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age',

for i in lst:

    if df[i].apply(type).nunique() > 1:

        print(f"Column '{i}' contains mixed types. Handling required.")

        df[i] = df[i].astype(str)

    df[i] = le.fit_transform(df[i])
```

```
Column 'Glucose' contains mixed types. Handling required.
Column 'BloodPressure' contains mixed types. Handling required.
Column 'SkinThickness' contains mixed types. Handling required.
Column 'BMI' contains mixed types. Handling required.
Column 'DiabetesPedigreeFunction' contains mixed types. Handling required.
Column 'Age' contains mixed types. Handling required.
Column 'Outcome' contains mixed types. Handling required.
```

```
df.dtypes
```

```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                         int64
DiabetesPedigreeFunction    int64
Age                         int64
Outcome                     int64
dtype: object
```

```
X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

```
from sklearn.preprocessing import MinMaxScaler
minmax=MinMaxScaler()
X=minmax.fit_transform(X)
X
```

```
array([[1.        , 1.        , 1.        , ..., 1.        , 1.        ,
        1.        ],
       [0.        , 0.98039216, 0.        , ..., 0.31410407, 0.61111111,
        0.16666667],
       [0.        , 0.69607843, 0.        , ..., 0.36425712, 0.61111111,
        0.83333333],
       ...,
       [0.33333333, 0.82352941, 0.        , ..., 0.3762656 , 0.27777778,
        0.27777778],
       [0.        , 0.37254902, 0.        , ..., 0.55497999, 0.05555556,
        0.        ],
       [0.        , 0.7254902 , 0.        , ..., 0.24911702, 0.61111111,
        0.94444444]])
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=1)
```

```
models = {
    'Random Forest': RandomForestClassifier(),
    'AdaBoost': AdaBoostClassifier(),
    'K-Neighbors': KNeighborsClassifier(),
    #'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeClassifier()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)

    accuracy = accuracy_score(y_test, predictions)
    report = classification_report(y_test, predictions)
    cm = confusion_matrix(y_test, predictions)

    print(f"{name}:")
    print(f"  Accuracy Score: {accuracy}")
    print("  Classification Report:")
    print(report)
    print("  Confusion Matrix:")
    print(cm)
    print()
```

```
           1       0.94      0.67      0.79      2488

    accuracy                           0.97     30001
   macro avg       0.96      0.84      0.89     30001
weighted avg       0.97      0.97      0.97     30001

  Confusion Matrix:
[[27415    98]
 [  812  1676]]
```

```
           1       1.00      0.66      0.80      2488

    accuracy                           0.97     30001
   macro avg       0.99      0.83      0.89     30001
weighted avg       0.97      0.97      0.97     30001

  Confusion Matrix:
[[27513     0]
 [  846  1642]]

K-Neighbors:
  Accuracy Score: 0.9570680977300756
  Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     27513
           1       0.87      0.56      0.69      2488

    accuracy                           0.96     30001
   macro avg       0.92      0.78      0.83     30001
weighted avg       0.95      0.96      0.95     30001

  Confusion Matrix:
[[27308   205]
 [ 1083  1405]]

Decision Tree:
  Accuracy Score: 0.9515682810572981
  Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.97     27513
           1       0.70      0.73      0.71      2488

    accuracy                           0.95     30001
   macro avg       0.84      0.85      0.84     30001
weighted avg       0.95      0.95      0.95     30001

  Confusion Matrix:
[[26742   771]
 [  682  1806]]
```