NAAN MUDHALVAN

FINAL DEVELOPMENT:

diabetes_predictions:

The **diabetes_prediction_dataset.csv file** contains medical and demographic data of patients along with their diabetes status, whether positive or negative. It consists of various features such as age, gender, body mass index (BMI), hypertension, heart disease, smoking history, HbA1c level, and blood glucose level. The Dataset can be utilized to construct machine learning models that can predict the likelihood of diabetes in patients based on their medical history and demographic details.

Prediction of diabetes

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.metrics import r2_score,confusion_matrix

# import warnings
import warnings
# filter warnings
warnings.filterwarnings('ignore')

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will
list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/diabetes-prediction-dataset/diabetes_prediction_dataset.csv
```

Load and Check Data

In [2]:

```
data = pd.read_csv("/kaggle/input/diabetes-prediction-dataset/diabetes_prediction_dataset.csv")
```

In [3]:

```
data.head()
```

Out[3]:

	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

In [4]:

```
data.info()
```

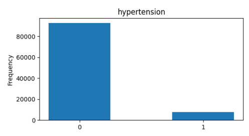
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   gender               100000 non-null  object 
 1   age                  100000 non-null  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
```

In [5]:

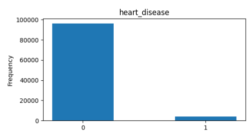
```
def bar_plot(variable):
    '''
    input: variable ex: "Sex"
    output: bar plot & value count
    '''
    # get feature
    var = data[variable]
    # count number of categorical variable(value/sample)
    var_value = var.value_counts()
    # visualize
    plt.figure(figsize=(6,3))
    plt.bar(var_value.index, var_value,width= 1/(var.unique().size))
    plt.xticks(var_value.index, var_value.index.values)
    plt.ylabel("Frequency")
    plt.title(variable)
    plt.show()
    print("{}: \n {}".format(variable,var_value))
```

In [6]:

```
category1 = ["hypertension","heart_disease","smoking_history"]
for c in category1:
    bar_plot(c)
```

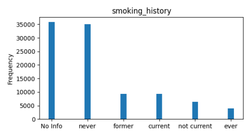


```
hypertension:
 hypertension
0    92515
1     7485
Name: count, dtype: int64
```



```
heart_disease:
 heart_disease
0    96058
```

```
1     3942
Name: count, dtype: int64
```



```
smoking_history:
 smoking_history
No Info        35816
never          35095
former          9352
current         9286
not current     6447
ever            4004
Name: count, dtype: int64
```

Change the Data Type

Making data types integer

In [7]:

```
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
```

In [8]:

```
pd.unique(data.smoking_history)
```

Out[8]:

```
array(['never', 'No Info', 'current', 'former', 'ever', 'not current'],

      dtype=object)
```

In [9]:

```
pd.unique(data.gender)
```

Out[9]:

```
array(['Female', 'Male', 'Other'], dtype=object)
```

For changing objects to integer

In [10]:

```
def change_string_to_int(column):
    variables=pd.unique(data[column])
    for item in range(variables.size):
        data[column]=[item if each==variables[item] else each for each in
data[column]]
    return data[column]
```

In [11]:

```
data["gender"]=change_string_to_int("gender")
```

In [12]:

```
data["smoking_history"]=change_string_to_int("smoking_history")
```

In [13]:

```
data.head()
```

Out[13]:

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

In [14]:

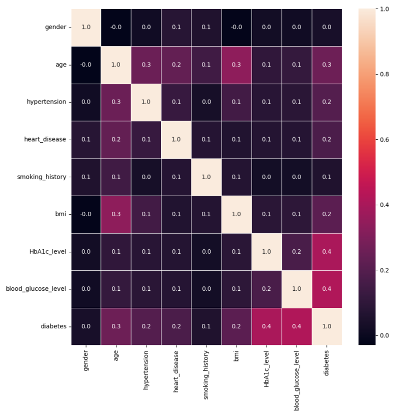
```
data.info()
```

```
<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  int64  
 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  int64  
 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(6)
memory usage: 6.9 MB
```

Removing from the Data

In [15]:

```
f,ax = plt.subplots(figsize=(10, 10))
sns.heatmap(data.corr(numeric_only=True), annot=True, linewidths=.5, fmt=
'.1f',ax=ax,)
plt.show()
```



In [16]:

```
data.drop("gender",axis=1,inplace=True)
```

In [17]:

```
data.head()
```

Out[17]:

	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	80.0	0	1	0	25.19	6.6	140	0
1	54.0	0	0	1	27.32	6.6	80	0
2	28.0	0	0	0	27.32	5.7	158	0

3	36.0	0	0	2	23.45	5.0	155	0
4	76.0	1	1	2	20.14	4.8	155	0

Transform the Data

```
In [18]:
data.describe()
```

```
Out[18]:
```

	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
count	100000.000000	100000.00000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000
mean	41.885856	0.07485	0.039420	1.306950	27.320767	5.527507	138.058060	0.085000
std	22.516840	0.26315	0.194593	1.454501	6.636783	1.070672	40.708136	0.278883

min	0.080000	0.00000	0.000000	0.000000	10.010000	3.500000	80.000000	0.000000
25%	24.000000	0.00000	0.000000	0.000000	23.630000	4.800000	100.000000	0.000000
50%	43.000000	0.00000	0.000000	1.000000	27.320000	5.800000	140.000000	0.000000
75%	60.000000	0.00000	0.000000	2.000000	29.580000	6.200000	159.000000	0.000000
max	80.000000	1.00000	1.000000	5.000000	95.690000	9.000000	300.000000	1.000000

Normalization

```
In [19]:
data = (data - data.min())/(data.max()-data.min())
```

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

```
Out[20]:
```

	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	1.000000	0.0	1.0	0.0	0.177171	0.563636	0.272727	0.0
1	0.674675	0.0	0.0	0.2	0.202031	0.563636	0.000000	0.0
2	0.349349	0.0	0.0	0.0	0.202031	0.400000	0.354545	0.0
3	0.449449	0.0	0.0	0.4	0.156863	0.272727	0.340909	0.0
4	0.949950	1.0	1.0	0.4	0.118231	0.236364	0.340909	0.0

In [21]:

```
data["hypertension"]= data["hypertension"].astype("int64")
data["heart_disease"]= data["heart_disease"].astype("int64")
data["diabetes"]= data["diabetes"].astype("int64")
```

In [22]:

```
data.head()
```

Out[22]:

	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	1.000000	0	1	0.0	0.177171	0.563636	0.272727	0
1	0.674675	0	0	0.2	0.202031	0.563636	0.000000	0
2	0.349349	0	0	0.0	0.202031	0.400000	0.354545	0
3	0.449449	0	0	0.4	0.156863	0.272727	0.340909	0
4	0.949950	1	1	0.4	0.118231	0.236364	0.340909	0

Split and Train

In [23]:

```
x = data.drop("diabetes",axis=1)
y = data.diabetes
```

In [24]:

```
x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size=0.3,random_state=42)
```

In [25]:

```
y_train = np.array(y_train).reshape(-1,1)
y_test = np.array(y_test).reshape(-1,1)
```

In [26]:

```
print("x train: ",x_train.shape)
print("x test: ",x_test.shape)
print("y train: ",y_train.shape)
print("y test: ",y_test.shape)
```

```
x train:  (70000, 7)
x test:  (30000, 7)
y train:  (70000, 1)
y test:  (30000, 1)
```

In [27]:

```
logreg = linear_model.LogisticRegression(random_state = 42,max_iter= 200,)
print("test accuracy: {} ".format(logreg.fit(x_train,
y_train).score(x_test, y_test)))
print("train accuracy: {} ".format(logreg.fit(x_train,
y_train).score(x_train, y_train)))
```

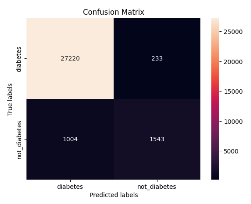
```
test accuracy: 0.9587666666666667
train accuracy: 0.9610571428571428
```

In [28]:

```
y_pred = logreg.fit(x_train, y_train).predict(x_test)
cm = confusion_matrix(y_test,y_pred)
```

In [29]:

```
ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
 ax.set_title('Confusion Matrix');
 ax.xaxis.set_ticklabels(['diabetes','not_diabetes']);
ax.yaxis.set_ticklabels(['diabetes','not_diabetes']);
```


Confusion Matrix
diabetes
not_diabetes
True labels
Predicted labels