Diabetes.csv

PA Consulting

Israt Ahmed

Must have

Data exploration

What are the types of your variable?

[1]:
```python
import pandas as pd
```

[3]:
```python
df_diabetes = pd.read_csv("diabetes.csv")
```

[4]:
```python
df_diabetes.dtypes
```

[4]:
```
id            int64
chol        float64
stab.glu      int64
hdl         float64
ratio       float64
glyhb       float64
age           int64
gender       object
height      float64
weight      float64
frame        object
bp.1s       float64
bp.1d       float64
waist       float64
hip         float64
dtype: object
```

What are the distributions and the summary statistics?

[9]:
```
df_diabetes.describe()
```

[9]:

	id	chol	stab.glu	hdl	ratio	glyhb	age	height	weight	bp.1s	bp.1d	wais
count	403.000000	402.000000	403.000000	402.000000	402.000000	390.000000	403.000000	398.000000	402.000000	398.000000	398.000000	401.00000
mean	15978.310174	207.845771	106.672457	50.445274	4.521642	5.589769	46.851117	66.020101	177.592040	136.904523	83.321608	37.90024
std	11881.122124	44.445557	53.076655	17.262626	1.727886	2.242595	16.312333	3.918515	40.340666	22.741033	13.589227	5.72931
min	1000.000000	78.000000	48.000000	12.000000	1.500000	2.680000	19.000000	52.000000	99.000000	90.000000	48.000000	26.00000
25%	4792.500000	179.000000	81.000000	38.000000	3.200000	4.380000	34.000000	63.000000	151.000000	121.250000	75.000000	33.00000
50%	15766.000000	204.000000	89.000000	46.000000	4.200000	4.840000	45.000000	66.000000	172.500000	136.000000	82.000000	37.00000
75%	20336.000000	230.000000	106.000000	59.000000	5.400000	5.600000	60.000000	69.000000	200.000000	146.750000	90.000000	41.00000
max	41756.000000	443.000000	385.000000	120.000000	19.299999	16.110001	92.000000	76.000000	325.000000	250.000000	124.000000	56.00000

Are there any missing values?

```
[10]: df_diabetes.isnull().sum()
```

```
[10]: id           0
      chol         1
      stab.glu     0
      hdl          1
      ratio        1
      glyhb       13
      age          0
      gender       0
      height       5
      weight       1
      frame       12
      bp.1s        5
      bp.1d        5
      waist        2
      hip          2
      dtype: int64
```

How are the variables correlated?

```
[16]: df_diabetes[["id","chol","stab.glu","hdl","ratio","glyhb","age","gender","height","weight","frame","bp.1s","bp.1d","waist","hip"]].corr
```

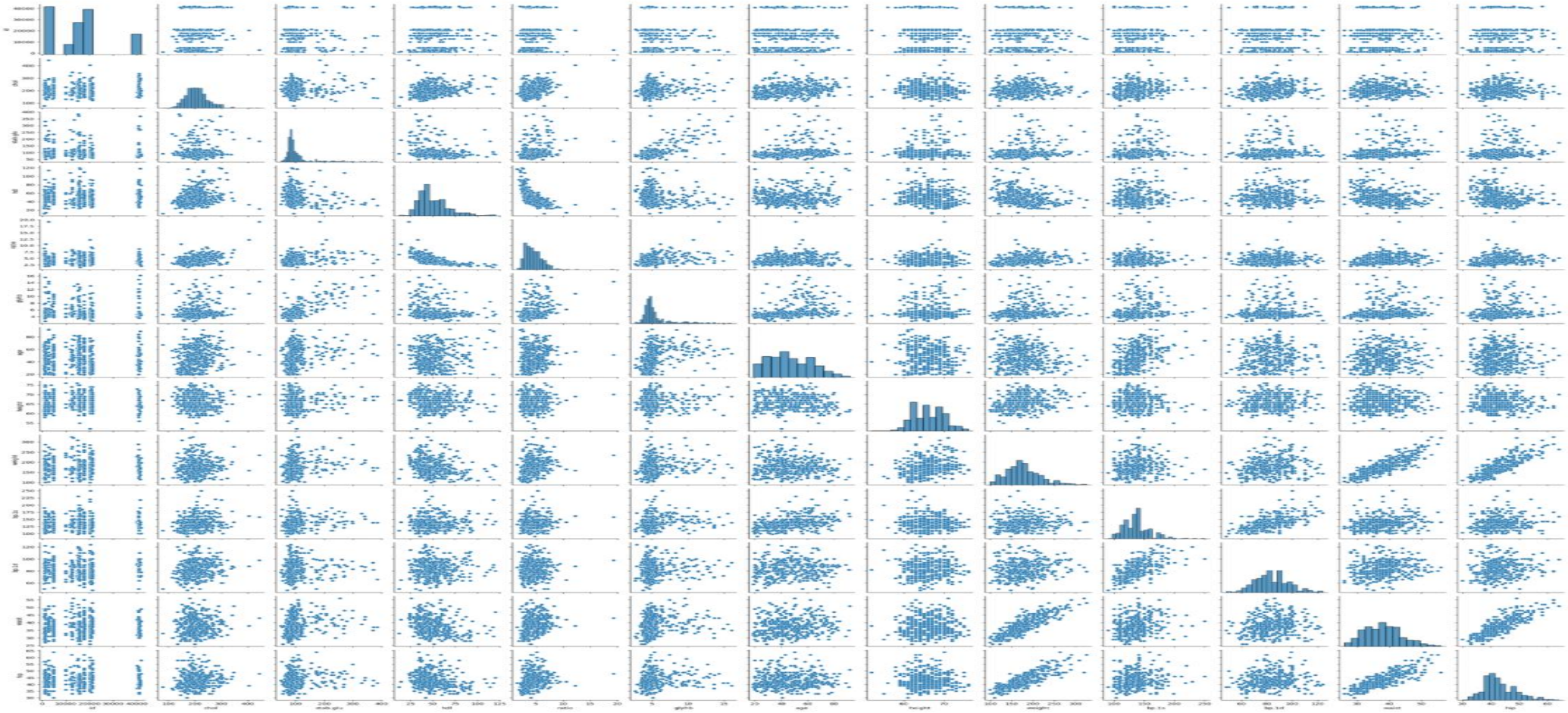
```
[16]: <bound method DataFrame.corr of         id   chol  stab.glu   hdl  ratio      glyhb  age  gender  height  \
0     1000  203.0        82  56.0    3.6   4.310000   46  female    62.0
1     1001  165.0        97  24.0    6.9   4.440000   29  female    64.0
2     1002  228.0        92  37.0    6.2   4.640000   58  female    61.0
3     1003   78.0        93  12.0    6.5   4.630000   67    male    67.0
4     1005  249.0        90  28.0    8.9   7.720000   64    male    68.0
..     ...    ...       ...   ...    ...        ...  ...     ...     ...
398  41506  296.0       369  46.0    6.4  16.110001   53    male    69.0
399  41507  284.0        89  54.0    5.3   4.390000   51  female    63.0
400  41510  194.0       269  38.0    5.1  13.630000   29  female    69.0
401  41752  199.0        76  52.0    3.8   4.490000   41  female    63.0
402  41756  159.0        88  79.0    2.0        NaN   68  female    64.0

     weight   frame  bp.1s  bp.1d  waist   hip
0     121.0  medium  118.0   59.0   29.0  38.0
1     218.0   large  112.0   68.0   46.0  48.0
2     256.0   large  190.0   92.0   49.0  57.0
3     119.0   large  110.0   50.0   33.0  38.0
4     183.0  medium  138.0   80.0   44.0  41.0
..      ...     ...    ...    ...    ...   ...
398   173.0  medium  138.0   94.0   35.0  39.0
399   154.0  medium  140.0  100.0   32.0  43.0
400   167.0   small  120.0   70.0   33.0  40.0
401   197.0  medium  120.0   78.0   41.0  48.0
402   220.0  medium  100.0   72.0   49.0  58.0

[403 rows x 15 columns]>
```

Visualise the relationship of your variables

```
[5]: import seaborn as sns
[8]: sns.pairplot(df_diabetes)
[8]: <seaborn.axisgrid.PairGrid at 0x1183309d0>
```



Data pre-processing

Do you need to convert any variables to the right type?

- Gender
- Frame

- Both from **Objects** to **Floats**

Do you need to encode categorical variables?

[47]:
```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder()
encoder.fit(X_train[['gender']])

gender_train_encoded = encoder.transform(X_train[['gender']])
df_gender_train = pd.DataFrame(gender_train_encoded.toarray(), columns = encoder.get_feature_names_out(['gender']))
X_train.reset_index(drop=True, inplace=True)
X_train = pd.concat([X_train, df_gender_train], axis=1).drop(['gender'], axis = 1)

gender_test_encoded = encoder.transform(X_test[['gender']])
df_gender_test = pd.DataFrame(gender_test_encoded.toarray(), columns = encoder.get_feature_names_out(['gender']))
X_test.reset_index(drop=True, inplace=True)
X_test = pd.concat([X_test, df_gender_test], axis=1).drop(['gender'], axis = 1)
```

[16]:
```
encoder.fit(X_train[['frame']])

frame_train_encoded = encoder.transform(X_train[['frame']])
df_frame_train = pd.DataFrame(frame_train_encoded.toarray(), columns = encoder.get_feature_names_out(['frame']))
X_train.reset_index(drop=True, inplace=True)
X_train = pd.concat([X_train, df_frame_train], axis=1).drop(['frame'], axis = 1)
X_train.head()

frame_test_encoded = encoder.transform(X_test[['frame']])
df_frame_test = pd.DataFrame(frame_test_encoded.toarray(), columns = encoder.get_feature_names_out(['frame']))
X_test.reset_index(drop=True, inplace=True)
X_test = pd.concat([X_test, df_frame_test], axis=1).drop(['frame'], axis = 1)
```

Do you need to remove/input missing data?

```
[45]: df_diabetes = df_diabetes.dropna(axis=0)
```

```
[46]: df_diabetes = df_diabetes.fillna(0)
```

Split your data into a training and a test set

[6]:
```
from sklearn.model_selection import train_test_split
```

[8]:
```
X = df_diabetes.drop(columns=["stab.glu"])
```

[9]:
```
y = df_diabetes["stab.glu"]
```

[10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

Build your regression model

Which variables are you going to include?

[20]:
```
model = LogisticRegression()
```

[21]:
```
model.fit(X_train, y_train)
```

```
/opt/anaconda3/envs/wit-python-ds/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to con
verge (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(
```

[21]:
```
▾ LogisticRegression
LogisticRegression()
```

What will be the functional form (relationship between variables) of your regression?

```
[22]: y_predictions = model.predict(X_test)
```

Fit your model on the training data and estimate the parameters

[22]:
```
y_predictions = model.predict(X_test)
```

[23]:
```
model.score(X_test,y_test)
```

[23]:
```
0.0
```

Evaluate your model and report your results

How well does your model [generalise]?

- Consistent

Which are variables are important in predicting if someone has diabetes?

- All except **glyhb** and **frame**

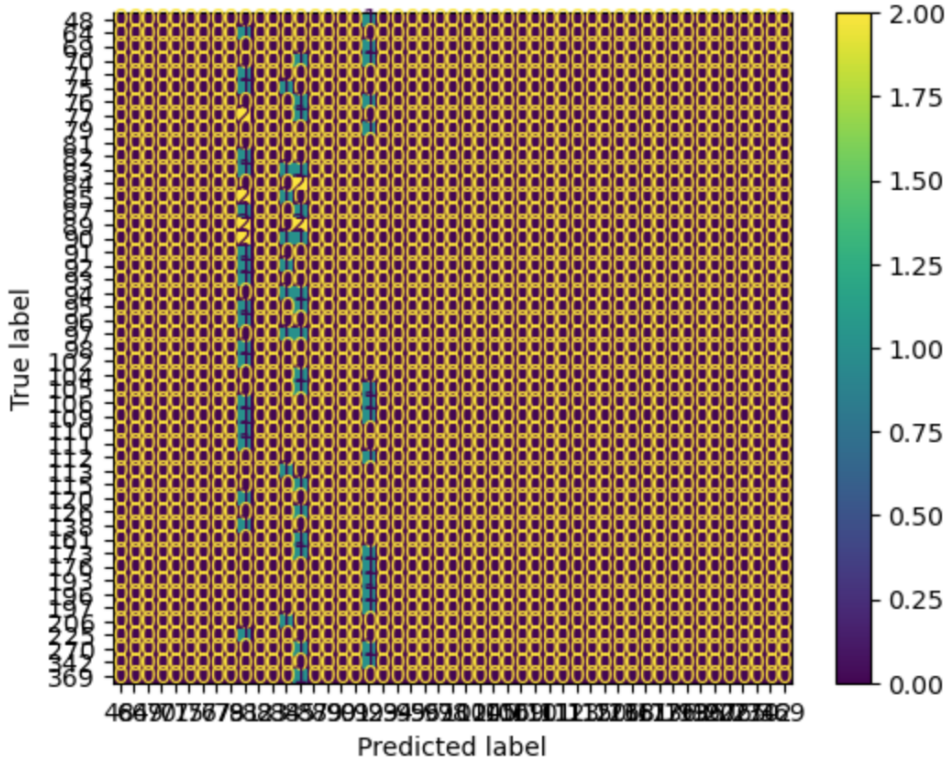
Are all regressors (variables) statistically significant?

Visualise your results to better understand the performance of your model

Draw a confusion Matrix

[26]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_predictions)
```

[26]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1290139a0>
```



[27]:
```
plt.show()
```