```
# importing the libraries

import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score

#loading the PIMA diabetes dataset
diabetes_dataset = pd.read_csv('/content/diabetes.csv')

#looking at the dataset
diabetes_dataset.head()
```

```
{"summary":"{\n  \"name\": \"diabetes_dataset\",\n  \"rows\": 768,\n
\"fields\": [\n    {\n      \"column\": \"Pregnancies\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 17,\n
\"num_unique_values\": 17,\n        \"samples\": [\n          6,\n
1,\n          3\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Glucose\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 31,\n        \"min\": 0,\n        \"max\": 199,\n
\"num_unique_values\": 136,\n        \"samples\": [\n          151,\n
101,\n          112\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BloodPressure\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 19,\n        \"min\": 0,\n
\"max\": 122,\n        \"num_unique_values\": 47,\n
\"samples\": [\n          86,\n          46,\n          85\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SkinThickness\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 15,\n        \"min\": 0,\n
\"max\": 99,\n        \"num_unique_values\": 51,\n        \"samples\":
[\n          7,\n          12,\n          48\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Insulin\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 115,\n
\"min\": 0,\n        \"max\": 846,\n        \"num_unique_values\":
186,\n        \"samples\": [\n          52,\n          41,\n
183\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"BMI\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7.8841603203754405,\n        \"min\": 0.0,\n        \"max\":
67.1,\n        \"num_unique_values\": 248,\n        \"samples\": [\n
19.9,\n          31.0,\n          38.1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DiabetesPedigreeFunction\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.33132859501277484,\n        \"min\": 0.078,\n        \"max\": 2.42,\
n        \"num_unique_values\": 517,\n        \"samples\": [\n
1.731,\n          0.426,\n          0.138\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 11,\n        \"min\": 21,\n
\"max\": 81,\n        \"num_unique_values\": 52,\n        \"samples\":
[\n          60,\n          47,\n          72\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Outcome\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          0,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"diabetes_dataset"}
```

```
# fetching number of rows and columns in our dataset
diabetes_dataset.shape
```

```
(768, 9)
```

```
diabetes_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

As we can observe, there are no missing values in the dataset and all the columns are numeric. Thus there is no issue since ml models work with numbers and not strings.

```
diabetes_dataset.describe() #statistical information
```

```
{"summary":"{\n  \"name\": \"diabetes_dataset\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"Pregnancies\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
269.85223453356366,\n        \"min\": 0.0,\n        \"max\": 768.0,\n
```

```
\"num_unique_values\": 8,\n        \"samples\": [\n
3.8450520833333335,\n          3.0,\n          768.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Glucose\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
243.73802348295857,\n        \"min\": 0.0,\n        \"max\": 768.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
120.89453125,\n          117.0,\n          768.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"BloodPressure\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
252.85250535810619,\n        \"min\": 0.0,\n        \"max\": 768.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
69.10546875,\n          72.0,\n          768.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SkinThickness\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
263.7684730531098,\n        \"min\": 0.0,\n        \"max\": 768.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          768.0,\n
20.536458333333332,\n          32.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Insulin\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
350.26059167945886,\n        \"min\": 0.0,\n        \"max\": 846.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          768.0,\n
79.79947916666667,\n          127.25\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"BMI\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 262.05117817552093,\n
\"min\": 0.0,\n        \"max\": 768.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          31.992578124999998,\n
32.0,\n          768.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"DiabetesPedigreeFunction\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 271.3005221658502,\n
\"min\": 0.078,\n        \"max\": 768.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
0.4718763020833325,\n          0.3725,\n          768.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 260.1941178528413,\n
\"min\": 11.76023154067868,\n        \"max\": 768.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
33.240885416666664,\n          29.0,\n          768.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Outcome\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
271.3865920388932,\n        \"min\": 0.0,\n        \"max\": 768.0,\n
\"num_unique_values\": 5,\n        \"samples\": [\n
0.34895833333333333,\n          1.0,\n          0.4769513772427971\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```

```
diabetes_dataset['Outcome'].value_counts()
```

```
Outcome
0    500
1    268
Name: count, dtype: int64
```

This shows that 500 patients do not have diabetes while 268 patients have diabetes.

0 ---> Not diabetic, 1 ---> Diabetic

```
diabetes_dataset.groupby('Outcome').mean()
```

```
{"summary":"{\n  \"name\": \"diabetes_dataset\",\n  \"rows\": 2,\n
\"fields\": [\n    {\n      \"column\": \"Outcome\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Pregnancies\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.108511248584296,\n        \"min\":
3.298,\n        \"max\": 4.865671641791045,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
4.865671641791045,\n          3.298\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Glucose\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
22.116505963980842,\n        \"min\": 109.98,\n        \"max\":
141.25746268656715,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          141.25746268656715,\n          109.98\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"BloodPressure\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.8672051632998017,\n        \"min\": 68.184,\n        \"max\":
70.82462686567165,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          70.82462686567165,\n          68.184\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"SkinThickness\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.7678935989570275,\n        \"min\": 19.664,\n        \"max\":
22.16417910447761,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          22.16417910447761,\n          19.664\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Insulin\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
```

```
22.304849659757796,\n        \"min\": 68.792,\n        \"max\":
100.33582089552239,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          100.33582089552239,\n          68.792\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"BMI\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 3.421221123996261 8,\
n        \"min\": 30.3042,\n        \"max\": 35.14253731343284,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
35.14253731343284,\n          30.3042\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"DiabetesPedigreeFunction\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.08539445753677459,\n        \"min\": 0.429734,\n        \"max\":
0.5505,\n        \"num_unique_values\": 2,\n        \"samples\": [\n
0.5505,\n          0.429734\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Age\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 4.155782645191446,\n        \"min\":
31.19,\n        \"max\": 37.06716417910448,\n
\"num_unique_values\": 2,\n        \"samples\": [\n
37.06716417910448,\n          31.19\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

Some useful insights from this outcome could be that those people who have diabetes have high amounts of glucose and blood pressure and also as we can see that the age group which is in the range of 37 or above or to be precise age group in the range of 35 and above are susceptible to diabetes whereas the younger age groups are not.

```
# splitting the dataset into training data and testing data
X=diabetes_dataset.drop('Outcome',axis=1)
y=diabetes_dataset['Outcome']

print(X)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
0              6      148             72             35        0  33.6

1              1       85             66             29        0  26.6

2              8      183             64              0        0  23.3

3              1       89             66             23       94  28.1

4              0      137             40             35      168  43.1

..           ...      ...            ...            ...      ...   ...

763           10      101             76             48      180  32.9
```

```
764                 2      122                70                27                0  36.8

765                 5      121                72                23              112  26.2

766                 1      126                60                 0                0  30.1

767                 1       93                70                31                0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
```

```
print(y)
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

As we can see, the features here have different ranges. With pregnancies ranging in single digits to glucose levels ranging in hundreds and this is the same with other columns as well. Thus we need to scale all of them down to a particular range since this will help us in giving more accurate results later on. Therefore we perform standardization next

```
# Standardization

scaler = StandardScaler()

#fitting the data
```

```
scaler.fit(X)

#transforming the features data
scaled_data=scaler.transform(X)

print(scaled_data)

[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]

X= scaled_data

print(X)

[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
```

Train Test Split

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,strat
ify=2,random_state=2)
```

Training the data

```
classifier = svm.SVC(kernel='linear',gamma='auto')

# training the svm classifier
classifier.fit(X_train,y_train)
```

```
SVC(gamma='auto', kernel='linear')
```

Model Evaluation

```
# accuracy score of the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction,y_train)
print(f'Accuracy score of the training data is:
{training_data_accuracy*100}%')
```

```
Accuracy score of the training data is:77.19869706840392%
```

```
# accuracy score of the testing data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction,y_test)
print(f'Accuracy score of the test data is:{test_data_accuracy*100}%')
```

```
Accuracy score of the test data is:76.62337662337663%
```

Making a Predictive System

```
# for this example..the output is 0. thus we need to test it.
input_data=(4,110,92,0,0,37.6,0.191,30)

#converting it into np array
input_data_np=np.asarray(input_data)

#reshaping it so that the model knows that we only want to know output
of one input.
input_data_reshaped=np.reshape(input_data_np,(1,-1))

# standardize it
std_data=scaler.transform(input_data_reshaped)
print(std_data)

#check the output
prediction=classifier.predict(std_data)
print(prediction)

if(prediction[0]==0):
  print("The patient is non-diabetic")
else:
  print("The patient is diabetic")
```

```
[[ 0.04601433 -0.34096773  1.18359575 -1.28821221 -0.69289057
0.71168975
  -0.84827977 -0.27575966]]
[0]
The patient is non-diabetic
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493:
UserWarning: X does not have valid feature names, but StandardScaler
was fitted with feature names
  warnings.warn(
```