Experiment. No. 10

Title : Implement K-nearest algorithm on diabetes.csv dataset compute confusion matrix, accuracy, error rate, precision and Recall on the given dataset.

objective : To preprocess dataset and identify outliers. to check correlation and implement KNN algorithm and Random Forest classification models. Evaluates them with respective scores like confusion_matrix, accuracy score, mean_squared_error, r2_score, roc_curve, etc

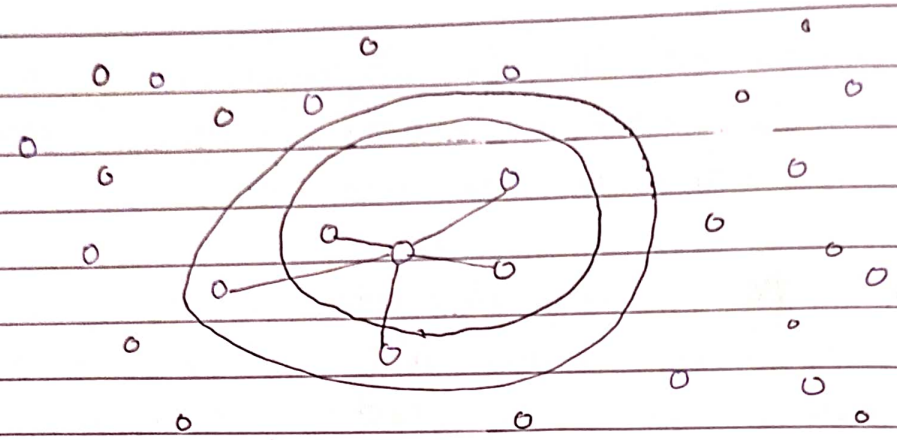
Dataset Description : We will try to build a ML model to accurately predict whether or not the patients in the dataset have diabetes or not?
The dataset consists of several medical predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age and so on.

Theory -

KNN :

K nearest Neighbors (KNN) is a supervised machine learning model. supervised learning is when a model learns from data is already labeled. A supervised learning models takes in a set of input objects and output values. the model then trains on data to learn how to inputs to the desired output so it can learn to make predictions on unseen data.

KNN model work by taking a data point and looking at the 'K' closest labelled data points.



For example : If k=5 and 3 of points are green and 2 are red then the data point in question would be labeled green since green is majority.

scikit-learn is a machine learning library for Python.

For KNN model, the first step is to read in the data we will use as input.

For this example, we are using the diabetes data To short, we will use pandas to read in data

Pregnancies	glucose	diastolic	triceps	insullin	bmi	dpf	age	
6	148	72	35	0	33	0.6	50	
1	85	66	29	0	26	0.3	31	0
8	183	64	0	0	23	0.6	32	1
1	89	66	23	23	28	6.1	21	0
0	137	40	35	35	48	0.2	33	1

Next step is see how much data we have, we will call shape function on our dataframe to see how many rows and columns there are in our data.

df.shape

op → (768, 9

split up the dataset into inputs and targets

```
x = df.drop(columns = ['diabetes'])
x.head()
```

we will insert the diabetes column of our dataset into our target variable (y)

```
y = df['diabetes'].values
y[0:5]
array([1,0,1,0,1])
```

split the dataset into train and test data

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split
    (x, y, test_size = 0.2, random_state = 1, startify = y)
```

Building and training model

```
knn = KNeighborsclassifier(n_neighbors)
knn.Fit(x_train, y_train)
```

First create new KNN classifier & set 'n_neighbors' to 3
we have to set 'n_neighbors' to 3 as a starting point
Next train the model using 'Fit' function and pass in our training data as parameters to fit our model to the training data

Testing model -

once model get trained, use 'predict' function on our model to make predictions on our test data.

```
y = earlier
0 = does not have diabetes
1 = have diabetes
show.dataset
knn.predict(x_test)[0:5]
=> array([0, 0, 0, 1])
```

For first four patients, showing no diabetes in data & has diabetes in test data & has diabetes 5th ~~dataset~~ patient.

Now let's see how accurate our model.

knn_score (x_test, y_test)

=> 0.668831168

our model has accuracy of approximately 66.88%.

K-Fold Cross validation -

Cross validation is when the dataset is randomly split up to k-means groups.

one of the groups is used as the test set and are used as training set

The model is trained on the training set and scored on the test set.

Then process is repeated until each unique gro as been used as the test set.

Hypertuning model parameter using Gridsearch cv -

Hypertuning parameters is when you go through a process to find the optimal parameters for your model to improve accuracy.

In our case, we will use Gridsearch cv to find the optimal value for 'n_neighbors'.

Gridsearch cv works by training our model multip times on a range of parameters that we specify

conclusion :

In this way, we build a neural network-based classifier that can determine whether they will ~~leave~~ have diabetes or not.