Diabetes Classification Using KNN

Project Report

Prepared By :

- Yuvanna Bawa | 102003294
- Rahul Narang | 102053040

Table Of Contents

Project Overview

The purpose of this project is to use diagnostic measures to predict the onset of diabetes. The dataset is from The National Institute of Diabetes and Digestive and Kidney Diseases. Using KNN Algorithm, we diagnostically predicted whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.

The Dataset Used

The link to the dataset used is - https://www.kaggle.com/uciml/pima-indians-diabetes-database. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.
The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

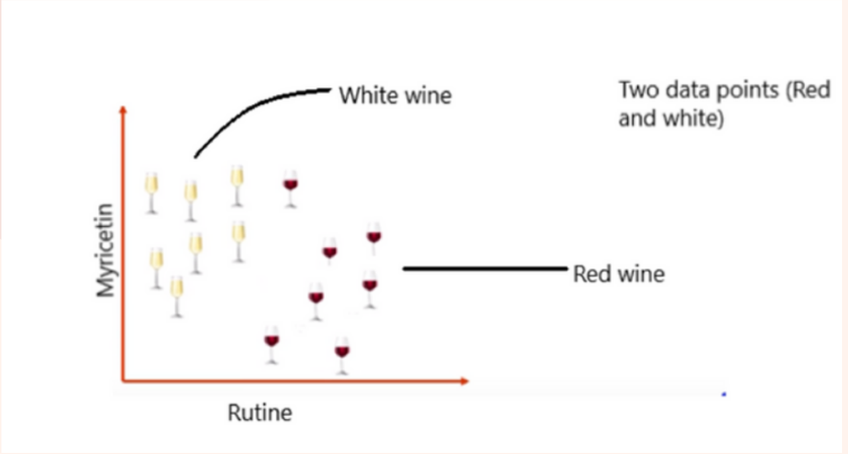
What is KNN?

KNN, or K Nearest Neighbours, is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure. It is mostly used to classifies a data point based on how its neighbours are classified.

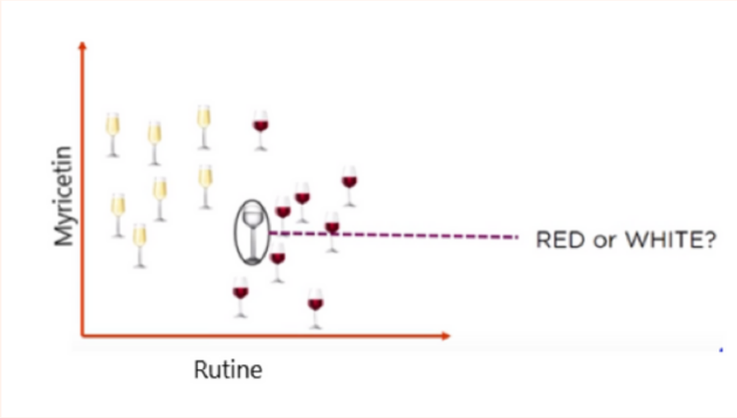
An example - let's take a wine example. Two chemical components called Rutime and Myricetin. Consider a measurement of Rutine vs Myricetin level with two data points, Red and White wines. They have tested and where then fall on that graph based on how much Rutine and how much Myricetin chemical content present in the wines.

What is KNN?

‘k’ in KNN is a parameter that refers to the number of nearest neighbours to include in the majority of the voting process.



Suppose, if we add a new glass of wine in the dataset. We would like to know whether the new wine is red or white?



So, we need to find out what the neighbours are in this case. Let’s say k = 5 and the new data point is classified by the majority of votes from its five neighbours and the new point would be classified as red since four out of five neighbours are red.

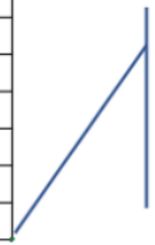
How KNN Works?

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given “unseen” observation. Similarity is defined according to a distance metric between two data points. A popular one is the Euclidean distance method

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Let’s take a small example. Age vs loan.

Customer	Age	Loan	Default
John	25	40000	N
Smith	35	60000	N
Alex	45	80000	N
Jade	20	20000	N
Kate	35	120000	N
Mark	52	18000	N
Anil	23	95000	Y
Pat	40	62000	Y
George	60	100000	Y
Jim	48	220000	Y
Jack	33	150000	Y
Andrew	48	142000	?



We need to predict Andrew default status by using Euclidean distance

How KNN Works?

We need to predict Andrew default status (Yes or No).

Calculate Euclidean distance for all the data points

Customer	Age	Loan	Default	Euclidean distance
John	25	40000	N	1,02,000.00
Smith	35	60000	N	82,000.00
Alex	45	80000	N	62,000.00
Jade	20	20000	N	1,22,000.00
Kate	35	120000	N	22,000.00
Mark	52	18000	N	1,24,000.00
Anil	23	95000	Y	47,000.01
Pat	40	62000	Y	80,000.00
George	60	100000	Y	42,000.00
Jim	48	220000	Y	78,000.00
Jack	33	150000	Y	8,000.01
Andrew	**48**	**142000**	**?**	

First Step calculate the Euclidean distance dist(d) = Sq.rt $(x_1-y_1)^2 + (x_2-y_2)^2$
= Sq.rt$(48-25)^2 + (142000 - 40000)^2$
dist (d_1) = 1,02,000.

We need to calcuate the distance for all the datapoints

With K=5, there are two Default=N and three Default=Y out of five closest neighbors. We can say default status for Andrew is ‘Y’ based on the major similarity of 3 points out of 5.

Customer	Age	Loan	Default	Euclidean distance	Minimum Euclidean Distance
John	25	40000	N	1,02,000.00	
Smith	35	60000	N	82,000.00	
Alex	**45**	**80000**	**N**	**62,000.00**	**5**
Jade	20	20000	N	1,22,000.00	
Kate	**35**	**120000**	**N**	**22,000.00**	**2**
Mark	52	18000	N	1,24,000.00	
Anil	**23**	**95000**	**Y**	**47,000.01**	**4**
Pat	40	62000	Y	80,000.00	
George	**60**	**100000**	**Y**	**42,000.00**	**3**
Jim	48	220000	Y	78,000.00	
Jack	**33**	**150000**	**Y**	**8,000.01**	**1**
Andrew	48	142000	?		

Let assume K = 5

Find minimum euclidean distance and rank in order (ascending)

In this case, 5 minimum euclidean distance. With k=5, there are two Default = N and three Default = Y out of five closest neighbors.

We can say Andrew default stauts is 'Y' (Yes)

How KNN Works?

Choosing K - What is K?

‘k’ in KNN algorithm is based on feature similarity choosing the right value of K is a process called parameter tuning and is important for better accuracy.

- To select the value of K that fits your data, we run the KNN algorithm multiple times with different K values. We’ll use accuracy as the metric for evaluating K performance. If the value of accuracy changes proportionally to the change in K, then it’s a good candidate for our K value.
- When it comes to choosing the best value for K, we must keep in mind the number of features and sample size per group. The more features and groups in our data set, the larger a selection we need to make in order to find an appropriate value of K.
- When we decrease the value of K to 1, our predictions become less stable. The accuracy decreases and the metric “F-Measure” becomes more sensitive to outliers. For better results, increase the value of K until the F-Measure value is higher than the threshold.

Understanding The Code

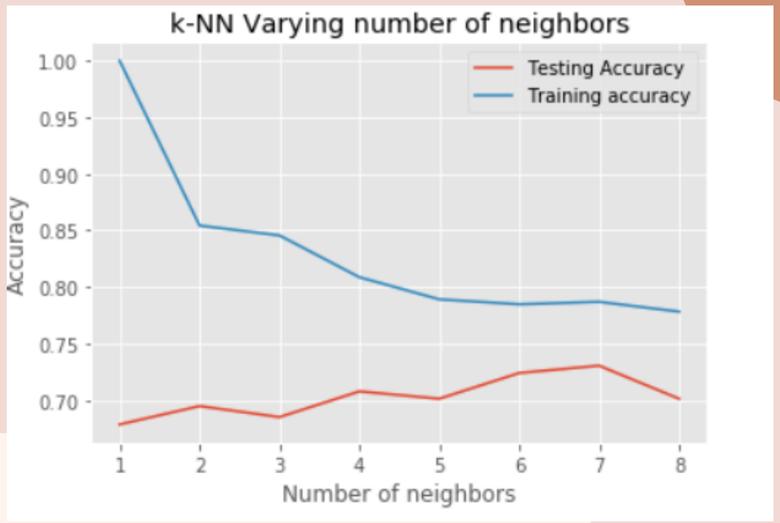
- X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.4,random_state=42, stratify=y)
  Splitting the dataset into train and test data, size of the test data being 40%

- knn = KNeighborsClassifier(n_neighbors=7) - Choosing value of k as 7, because at 7 the testing accuracy is maximum

The KNN Model

```
for i,k in enumerate(neighbors):
    #Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=k)

    #Fit the model
    knn.fit(X_train, y_train)
```



Understanding The Code

Confusion Matrix:

The confusion matrix is a technique used for summarizing the performance of a classification algorithm i.e. it has binary outputs

```
array([[165,  36],
       [ 47,  60]], dtype=int64)
```

Considering confusion matrix above:

True negative = 165

False positive = 36

True postive = 60

Fasle negative = 47

Understanding The Code

Classification Report:

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Precision = TP/TP+FP

Recall (Sensitivity) is the ratio of correctly predicted positive observations to the all observations in actual class - yes. The question recall answers is: Of all the passengers that truly survived, how many did we label? A recall greater than 0.5 is good.

Recall = TP/TP+FN

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall.

F1 Score = 2(Recall Precision) / (Recall + Precision)

Understanding The Code

Classification Report

Another important report is the Classification report. It is a text summary of the precision, recall, F1 score for each class.

```
#import classification_report
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test,y_pred))
```

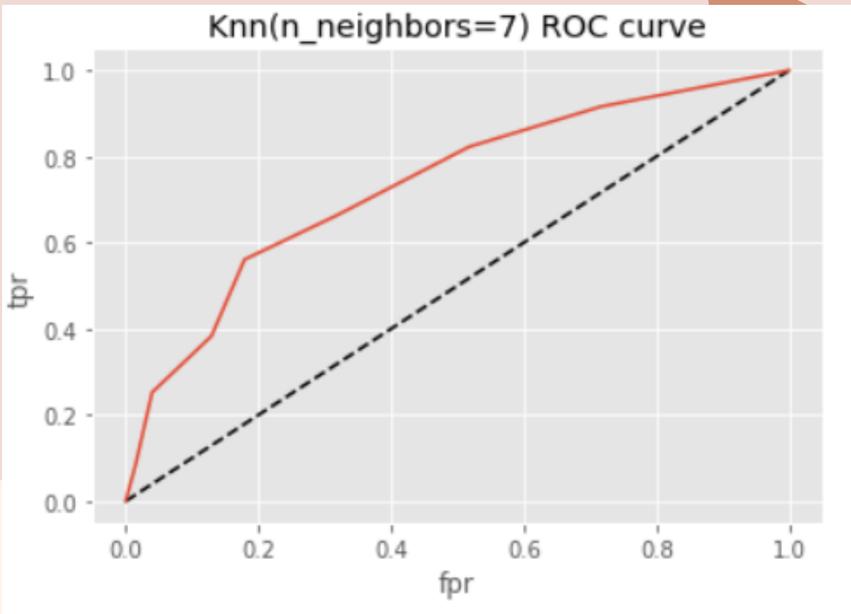
```
              precision    recall  f1-score   support

           0       0.78      0.82      0.80       201
           1       0.62      0.56      0.59       107

avg / total       0.73      0.73      0.73       308
```

ROC Curve :

ROC (Receiver Operating Characteristic) Curve tells us about how good the model can distinguish between two things (e.g If a patient has a disease or no). Better models can accurately distinguish between the two. Whereas, a poor model will have difficulties in distinguishing between the two.



Understanding The Code

Cross Validation :

When model is split into training and testing it can be possible that specific type of data point may go entirely into either training or testing portion. This would lead the model to perform poorly. Hence over-fitting and underfitting problems can be well avoided with cross validation techniques. Stratify parameter makes a split so that the proportion of values in the sample produced will be the same as the proportion of values provided to parameter stratify.

Understanding The Code

Hyperparameter Tuning :

Grid search is an approach to hyperparameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid.

```
#import GridSearchCV
from sklearn.model_selection import GridSearchCV
```

```
#In case of classifier like knn the parameter to be tuned is n_neighbors
param_grid = {'n_neighbors':np.arange(1,50)}
```

```
knn = KNeighborsClassifier()
knn_cv= GridSearchCV(knn,param_grid,cv=5)
knn_cv.fit(X,y)
```

```
GridSearchCV(cv=5, error_score='raise',
       estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform'),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

The Whole Project

https://github.com/YB73/AI-Project-Diabetes-Classifier-Using-KNN.git