PREDICT THE CATEGORY OF CUSTOMERS USING K-NEAREST NEIGHBOUR ALGORITHM

STREAM: M-TECH IN INFORMATION SECURITY

BY:

RITAM GHOSH

SAYANTA HARH

TANISHA GHOSHAL

QUAZI AHMED SAQUIB

ABSTRACT

KNN (K-Nearest Neighbour) is an extensively used classification algorithm owing to its simplicity, ease of implementation and effectiveness. It classifies according to its neighbours are classified. Classifiers are designed and developed to learn a model that maps the inputs to the class which are labelled as the training set and becomes available to perform and use. It basically predicts the output from the definite training sets. That's the reason it doesn't learn a discriminative function from the training set, rather it memorizes the data set. KNN has few shortcomings affecting its accuracy of classification. It has large memory requirements as well as high time complexity. We have experimentally tested our proposed algorithm in JUPYTER NOTEBOOK, using a standard Tele Customer dataset. The accuracy of our algorithm is improved with a blend of classification techniques. Experimental results have proved that our proposed algorithm performs better than conventional KNN algorithm.

TABLE OF CONTENTS

INTRODUCTION

CLASSIFICATION IN MACHINE LEARNING:

Classification in machine learning and statistics is a supervised learning approach in which the computer program learns from the data given to it and makes new observations or classifications.

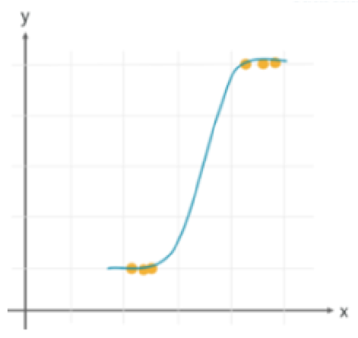
Classification is a process of ordering a given arrangement of data into classes. It can be performed on both structured and unstructured information. The process begins with predicting the class of given data points. The classes are regularly alluded to as target, name or classifications.

The classification predictive modelling is the task of approximating the mapping function from input variables to discrete output variables. The principle objective is to distinguish which class/classification the new information will fall into.

In machine learning, **classification** is a supervised learning concept which fundamentally orders a set of data into classes. The most common classification problems are – speech recognition, face detection, handwriting recognition, document classification, etc. It can be either a binary classification problem or a multi-class problem too. There are a lot of machinelearningalgorithms for classification in machine learning. Some algorithms are as follows:

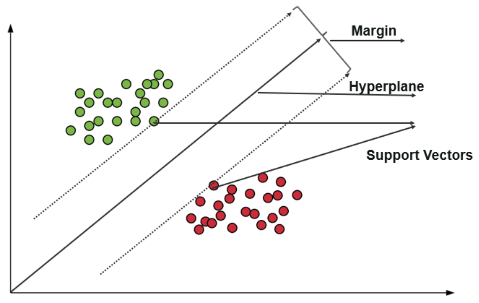
LOGISTIC REGRESSION:

It is a classification algorithm in machine learning that uses one or more independent variables to determine an outcome. The result is estimated with a dichotomous variable importance it will have just two potential results. The objective of logistic regression is to find a best-fitting relationship between the dependent variable and a set of independent variables.



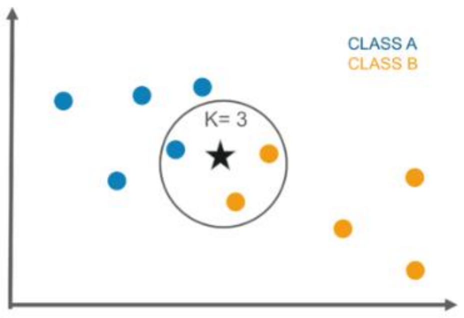
SUPPORT VECTOR MACHINE:

The support vector machine is a classifier that represents the training data as points in space separated into categories by a gap as wide as possible. New points are then added to space by predicting which category they fall into and which space they will belong to.



K-NEAREST NEIGHBOUR:

It is a lazy learning algorithm that stores all instances corresponding to training data in n-dimensional space**.** It is a lazy learning algorithm as it does not focus on constructing a general internal model; instead, it works on storing instances of training data.



(As the topic is based on K-Nearest Neighbour Algorithm we will broadly describe it here)

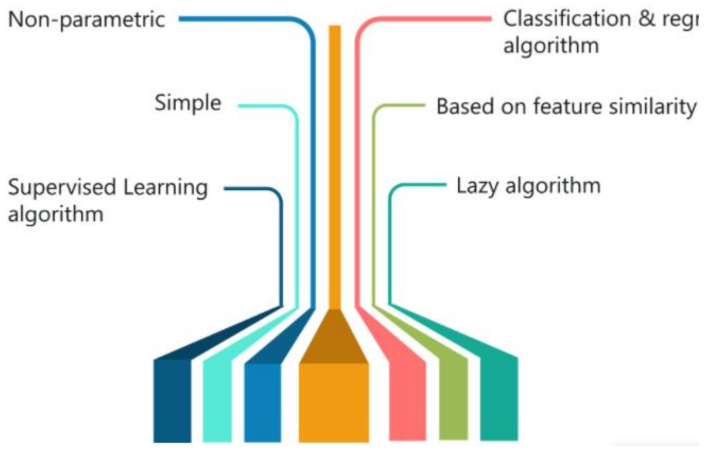
K-NEAREST NEIGHBOUR ALGORITHM:

DEFINITION:

KNN which stand for K Nearest Neighbour is a Supervised Machine Learning algorithm that classifies a new data point into the target class, depending on the features of its neighbouring data points.

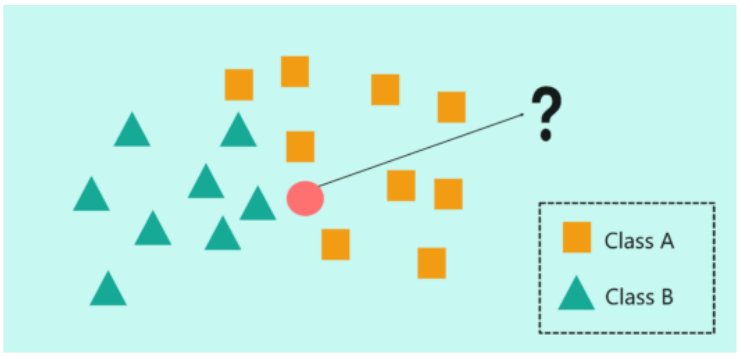
FEATURES:

- KNN is a Supervised Learning algorithm that uses labelled input data set to predict the output of the data points.
- It is one of the simplest Machine learning algorithms and it can be easily implemented for a varied set of problems.
- It is mainly based on feature similarity. KNN checks how similar a data point is to its neighbour and classifies the data point into the class it is most similar to.
- KNN is a non-parametric model which means that it does not make any assumptions about the data set.
- KNN is a lazy algorithm; this means that it memorizes the training data set instead of learning a discriminative function from the training data.
- Both classification and regression problems can be solved by KNN.

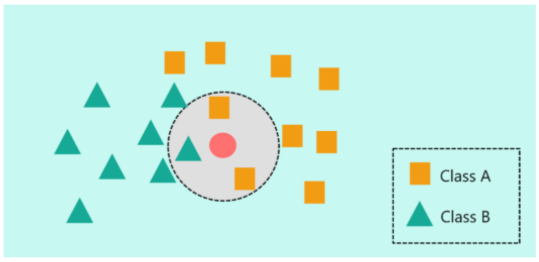

Non-parametric
Simple
Supervised Learning algorithm
Classification & regı algorithm
Based on feature similarity
Lazy algorithm

WORKING OF KNN ALGORITHM:

- In this image, we have two classes of data, namely Class A (squares) and Class B (triangles)
- The problem statement is to assign the new input data point to one of the two classes by using the KNN algorithm
- The first step in the KNN algorithm is to define the value of 'K' which stands for the number of Nearest Neighbours and hence the name K Nearest Neighbours (KNN).



- Here the value of 'K' is 3. This means that the algorithm will consider the three neighbours that are the closest to the new data point in order to decide the class of this new data point.
- At 'K' = 3, the neighbours include two squares and 1 triangle. So, the new data point based on 'K' = 3, then it would be assigned to Class A (squares).



- KNN uses Euclidean distance as a measure to check the distance between a new data point and its neighbours.

$$Dis(x_1, x_2) = \sqrt{\sum_{i=0}^{n} (x_{1i} - x_{2i})^2}$$

EVALUATION METRICS:

Evaluation matrix provides a key role to develop the model for accuracy. There are different types of evaluation metrics. We have used one of them.

F1 score:-

For f1 score there is a huge roles of confusion matrix. Here x axis represent the predicted value (y1) and y axis represent the actual value (y).

Here four sections will be divided.

a. TP:- True Positive (The value is positive and it is truly predicted)
b. TN:- True Negative (The value is negative and it also truly predicted)
c. FP:- False Positive (The value is positive and it is falsely predicted)
d. FN:- False Negative (The value is negative and it is falsely predicted)

To find the f1 score precision and recall should be calculated

$$Precision = \frac{TP}{(TP + FP)}$$

$$Recall = \frac{TP}{(TP + FN)}$$

$$F1 - score = \frac{2 \times (Precision + Recall)}{(Precision + Recall)}$$

Log Loss:-

Log loss performance of a classifier where the predicted output is a probability value between 0 and 1.

y: Actual value

y1: Predicted value

$$LogLoss = -\frac{1}{n}\sum(y \times \log(y1) + (1 - y) \times \log(1 - y1))$$

PRACTICAL IMPLEMENTION OF K-NN ALGORITHM IN PYTHON:

PROBLEM STATEMENT:

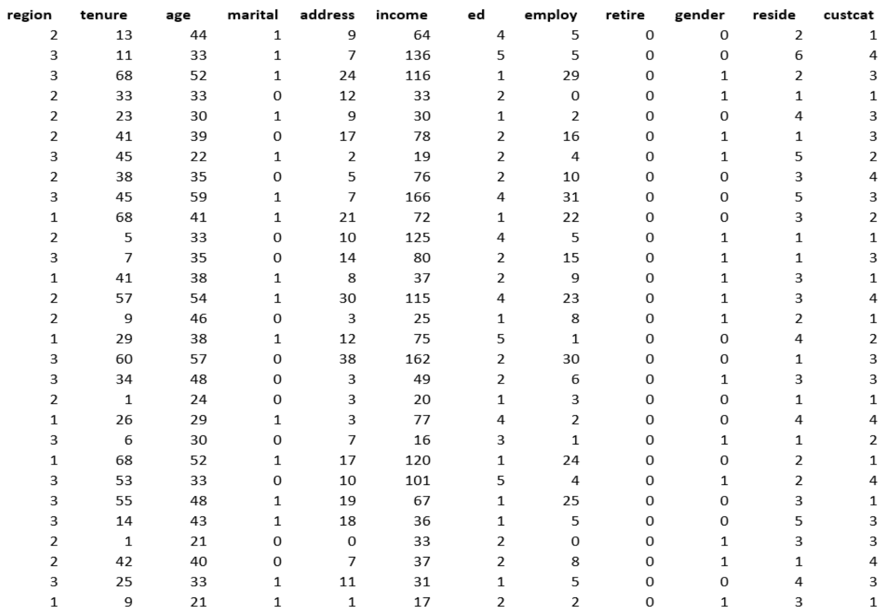
To study a customer dataset related to a telecommunication company, cleaning it and to predict the category of the customers and evaluate the accuracy of the machine learning model.

DATASET DESCRIPTION:

A telecommunications provider has segmented its customer base by service usage patterns, categorizing the customers into four groups. The given the dataset, with predefined labels, a model will be built to predict class of a new or unknown case.

Demographic data such as region, age, and marital, is used to predict usage patterns.

The target field, called custcat, has four possible values that correspond to the four customer groups, as follows: 1- Basic Service 2- E-Service 3- Plus Service 4- Total Service

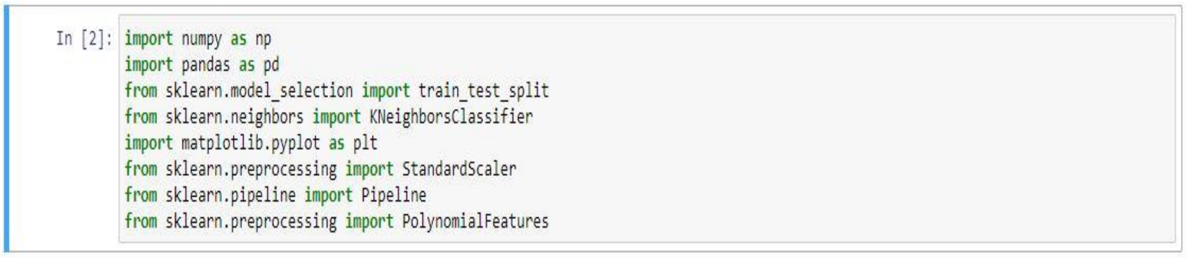


region	tenure	age	marital	address	income	ed	employ	retire	gender	reside	custcat
2	13	44	1	9	64	4	5	0	0	2	1
3	11	33	1	7	136	5	5	0	0	6	4
3	68	52	1	24	116	1	29	0	1	2	3
2	33	33	0	12	33	2	0	0	1	1	1
2	23	30	1	9	30	1	2	0	0	4	3
2	41	39	0	17	78	2	16	0	1	1	3
3	45	22	1	2	19	2	4	0	1	5	2
2	38	35	0	5	76	2	10	0	0	3	4
3	45	59	1	7	166	4	31	0	0	5	3
1	68	41	1	21	72	1	22	0	0	3	2
2	5	33	0	10	125	4	5	0	1	1	1
3	7	35	0	14	80	2	15	0	1	1	3
1	41	38	1	8	37	2	9	0	1	3	1
2	57	54	1	30	115	4	23	0	1	3	4
2	9	46	0	3	25	1	8	0	1	2	1
1	29	38	1	12	75	5	1	0	0	4	2
3	60	57	0	38	162	2	30	0	0	1	3
3	34	48	0	3	49	2	6	0	1	3	3
2	1	24	0	3	20	1	3	0	0	1	1
1	26	29	1	3	77	4	2	0	0	4	4
3	6	30	0	7	16	3	1	0	1	1	2
1	68	52	1	17	120	1	24	0	0	2	1
3	53	33	0	10	101	5	4	0	1	2	4
3	55	48	1	19	67	1	25	0	0	3	1
3	14	43	1	18	36	1	5	0	0	5	3
2	1	21	0	0	33	2	0	0	1	3	3
2	42	40	0	7	37	2	8	0	1	1	4
3	25	33	1	11	31	1	5	0	0	4	3
1	9	21	1	1	17	2	2	0	1	3	1

IMPLEMENTATION:

Now as the objective of the project is known let's get started.

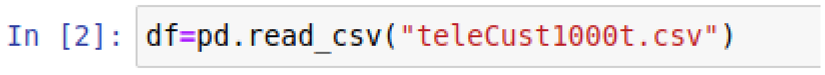
STEP 1: IMPORT THE LIBRARIES:



```
In [2]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.neighbors import KNeighborsClassifier
        import matplotlib.pyplot as plt
        from sklearn.preprocessing import StandardScaler
        from sklearn.pipeline import Pipeline
        from sklearn.preprocessing import PolynomialFeatures
```

- ➢ We have imported the Numpy library to convert our data into array form before fitting it to the algorithm to avoid shape issues.
- ➢ Pandas library lets us to manage, store and manipulate our dataset values.
- ➢ Train_test_split library enables us to divide our dataset into testing dataset and training datasets.
- ➢ KNeighborClassifier is our main library as it is the algorithm we used to get the result in this following project work.
- ➢ Matplotlib library is used to plot variety of graph for visual understanding of the datas present as well the generated output.
- ➢ StandardScalar and PolynomialFeatures libraries are used to scale down the values in our dataset to get an even better result. They are feature scaling tools.
- ➢ Pipeline library is used to simultaneously apply both the feature scaling operations to our datas.

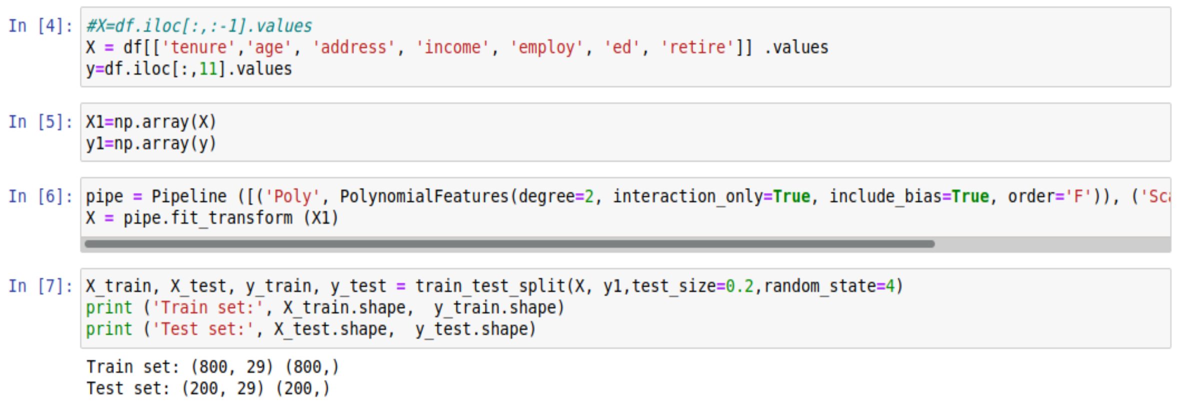
STEP 2: IMPORT THE DATASET:



```
In [2]: df=pd.read_csv("teleCust1000t.csv")
```

- ➢ The dataset was downloaded from canvas as provided by our mentor Dr. Somdatta Chakrabortty ma'am. We used our system path to store the dataset and we read the dataset from there itself.

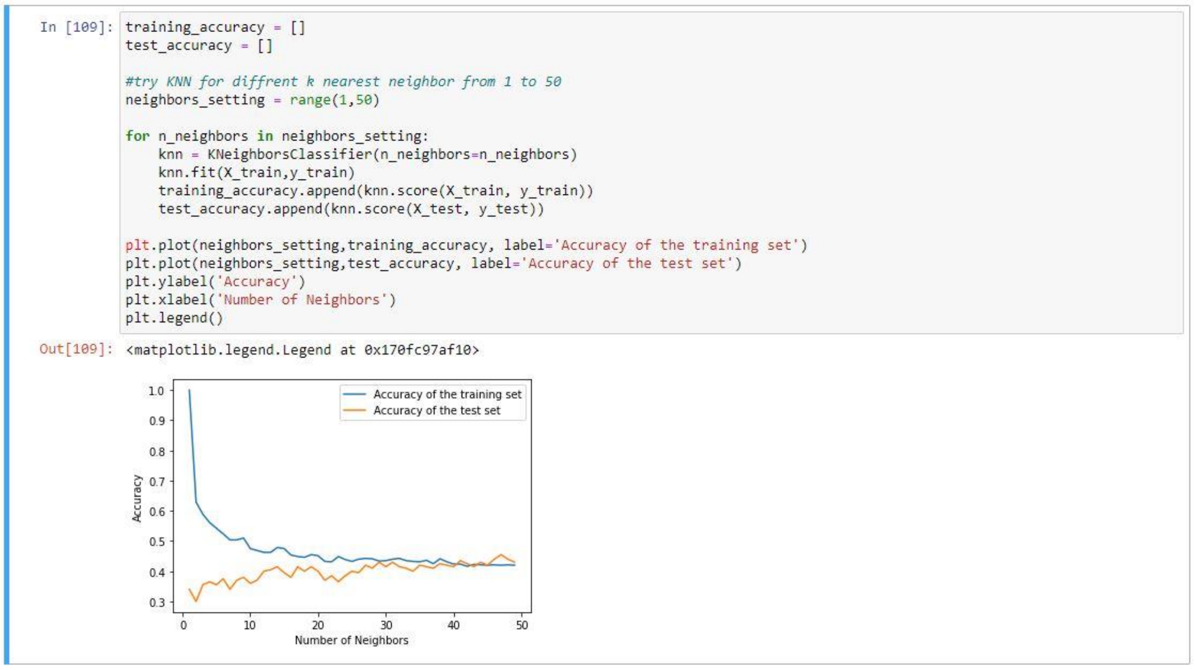
STEP 3: DATA CLEANING AND NORMALIZATION:



```
In [4]: #X=df.iloc[:,:-1].values
        X = df[['tenure','age', 'address', 'income', 'employ', 'ed', 'retire']] .values
        y=df.iloc[:,11].values

In [5]: X1=np.array(X)
        y1=np.array(y)

In [6]: pipe = Pipeline ([('Poly', PolynomialFeatures(degree=2, interaction_only=True, include_bias=True, order='F')), ('Sc
        X = pipe.fit_transform (X1)

In [7]: X_train, X_test, y_train, y_test = train_test_split(X, y1,test_size=0.2,random_state=4)
        print ('Train set:', X_train.shape,  y_train.shape)
        print ('Test set:', X_test.shape,  y_test.shape)

        Train set: (800, 29) (800,)
        Test set: (200, 29) (200,)
```

- ➢ The data was already cleaned and hard cleaning was not necessary on it. We simply let it run with Polynomial feature and Standard scaling feature scaling libraries to trim the value to lower limits to get better results.
- ➢ The next step was to put the data through test_train_split library to divide it into testing dataset and training dataset.

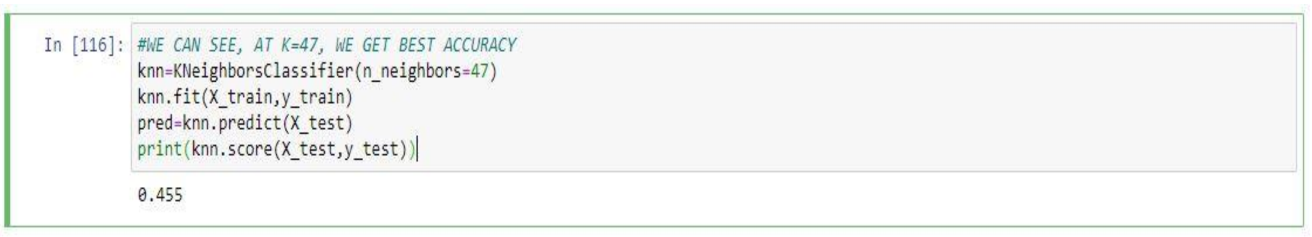
STEP 4: DESIGNING A MACHINE LEARNING MODEL:



```
In [109]: training_accuracy = []
          test_accuracy = []

          #try KNN for diffrent k nearest neighbor from 1 to 50
          neighbors_setting = range(1,50)

          for n_neighbors in neighbors_setting:
              knn = KNeighborsClassifier(n_neighbors=n_neighbors)
              knn.fit(X_train,y_train)
              training_accuracy.append(knn.score(X_train, y_train))
              test_accuracy.append(knn.score(X_test, y_test))

          plt.plot(neighbors_setting,training_accuracy, label='Accuracy of the training set')
          plt.plot(neighbors_setting,test_accuracy, label='Accuracy of the test set')
          plt.ylabel('Accuracy')
          plt.xlabel('Number of Neighbors')
          plt.legend()

Out[109]: <matplotlib.legend.Legend at 0x170fc97af10>
```

- ➢ Using KNN algorithm is all about getting the right value of K to get the best possible result. So we took a loop of values (1-50) and let it run as the value of K in our algorithm and plotted a graph to get the best possible value of K which is suitable for this particular problem.
- ➢ We found that when the value of K is 47, we get the highest accuracy which is 45.5%.
- ➢ The graph was plotted to get a better understanding of the testing and training accuracy with respect to the value K.

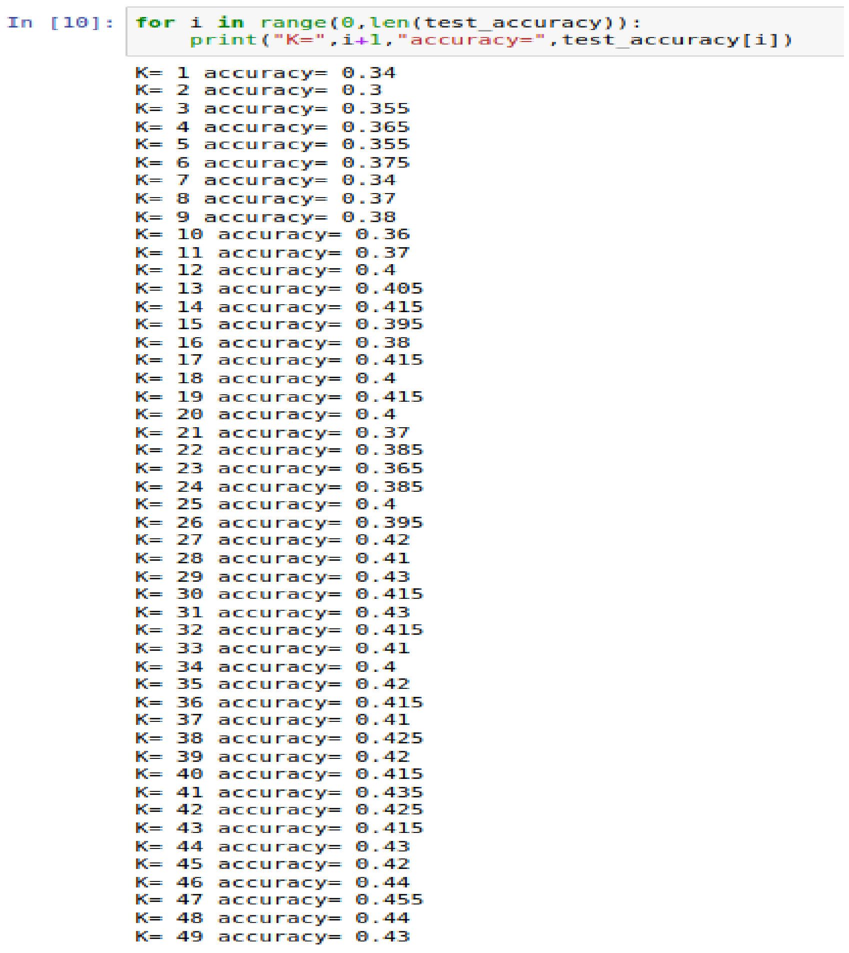
STEP 5: MODEL EVALUATION:



```
In [116]: #WE CAN SEE, AT K=47, WE GET BEST ACCURACY
          knn=KNeighborsClassifier(n_neighbors=47)
          knn.fit(X_train,y_train)
          pred=knn.predict(X_test)
          print(knn.score(X_test,y_test))

          0.455
```

- ➢ After confirming that using the value of K as 47 we get the best possible result for this problem, the next obvious step is implementing it. We implemented it to our algorithm to get ready to get the output.

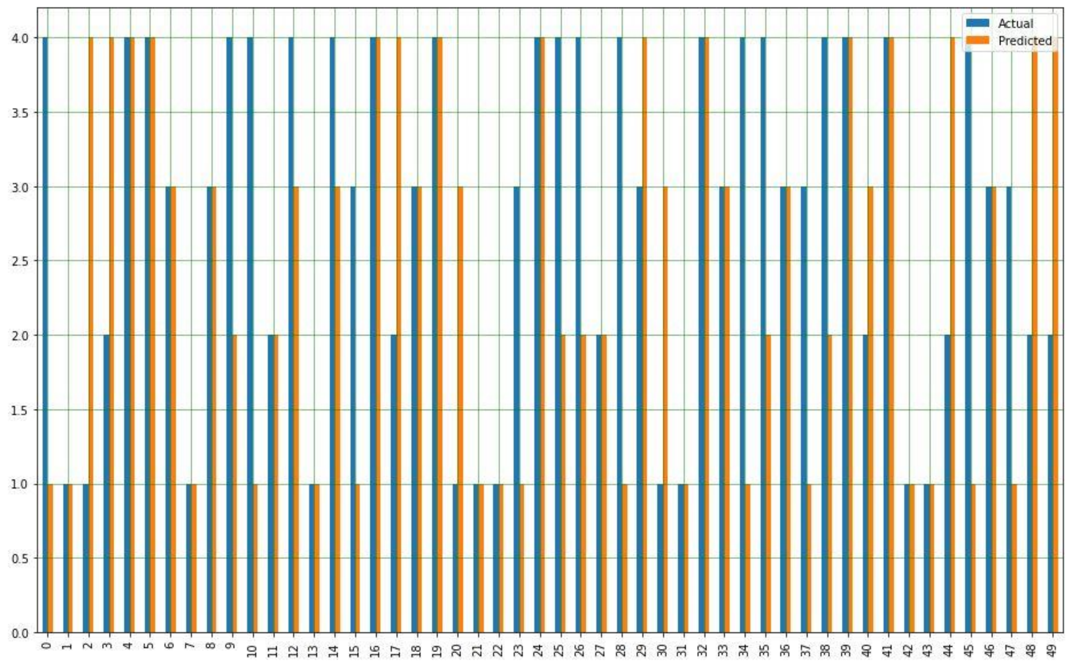
STEP 6: OPTIMIZATION:



```
In [10]: for i in range(0,len(test_accuracy)):
             print("K=",i+1,"accuracy=",test_accuracy[i])

         K= 1 accuracy= 0.34
         K= 2 accuracy= 0.3
         K= 3 accuracy= 0.355
         K= 4 accuracy= 0.365
         K= 5 accuracy= 0.355
         K= 6 accuracy= 0.375
         K= 7 accuracy= 0.34
         K= 8 accuracy= 0.37
         K= 9 accuracy= 0.38
         K= 10 accuracy= 0.36
         K= 11 accuracy= 0.37
         K= 12 accuracy= 0.4
         K= 13 accuracy= 0.405
         K= 14 accuracy= 0.415
         K= 15 accuracy= 0.395
         K= 16 accuracy= 0.38
         K= 17 accuracy= 0.415
         K= 18 accuracy= 0.4
         K= 19 accuracy= 0.415
         K= 20 accuracy= 0.4
         K= 21 accuracy= 0.37
         K= 22 accuracy= 0.385
         K= 23 accuracy= 0.365
         K= 24 accuracy= 0.385
         K= 25 accuracy= 0.4
         K= 26 accuracy= 0.395
         K= 27 accuracy= 0.42
         K= 28 accuracy= 0.41
         K= 29 accuracy= 0.43
         K= 30 accuracy= 0.415
         K= 31 accuracy= 0.43
         K= 32 accuracy= 0.415
         K= 33 accuracy= 0.41
         K= 34 accuracy= 0.4
         K= 35 accuracy= 0.42
         K= 36 accuracy= 0.415
         K= 37 accuracy= 0.41
         K= 38 accuracy= 0.425
         K= 39 accuracy= 0.42
         K= 40 accuracy= 0.415
         K= 41 accuracy= 0.435
         K= 42 accuracy= 0.425
         K= 43 accuracy= 0.415
         K= 44 accuracy= 0.43
         K= 45 accuracy= 0.42
         K= 46 accuracy= 0.44
         K= 47 accuracy= 0.455
         K= 48 accuracy= 0.44
         K= 49 accuracy= 0.43
```

- ➢ This output enabled us to better understand the variation of accuracy/ efficiency score of our algorithm with differing value of K
- ➢ We plotted a graph for it earlier but a more detailed printed form gave us the exact accuracy score.

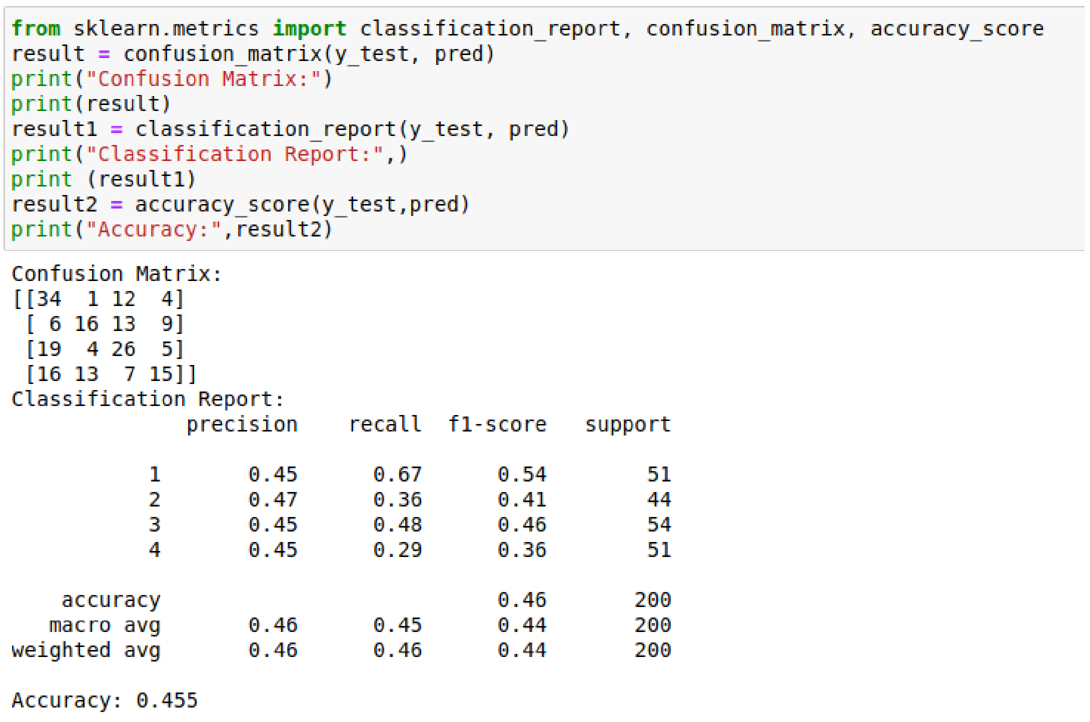
```
In [13]: out1.head(50).plot(kind='bar', figsize=(16,10))
         plt.grid(which='major', linestyle='-',linewidth='0.5', color='green')
         plt.grid(which='minor', linestyle=':',linewidth='0.5', color='black')
         plt.show()
```



- After our algorithm and data was fitted with right value of K, we used the predict method to get to compare the accuracy of our algorithm with the actual result. This is a comparison graph between the real output vs the obtained output.
- An accuracy of 45.5% is very low in real life, hence most of the datas predicted were inaccurate.

RESULT

F1-ACCURACY PREDICTION WITH APPROPRIATE K VALUE:



```python
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
result = confusion_matrix(y_test, pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, pred)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test,pred)
print("Accuracy:",result2)
```

```
Confusion Matrix:
[[34  1 12  4]
 [ 6 16 13  9]
 [19  4 26  5]
 [16 13  7 15]]
Classification Report:
              precision    recall  f1-score   support

           1       0.45      0.67      0.54        51
           2       0.47      0.36      0.41        44
           3       0.45      0.48      0.46        54
           4       0.45      0.29      0.36        51

    accuracy                           0.46       200
   macro avg       0.46      0.45      0.44       200
weighted avg       0.46      0.46      0.44       200

Accuracy: 0.455
```

CONCLUSION

K- Nearest Neighbours (KNN) identifies the nearest neighbours given the value of K. It is lazy learning and non-parametric algorithm. Here in the above experiment we achieve accuracy of 0.455 after at k = 47 after passing it through a range of values. So we can conclude that KNN works on low dimension dataset while faces problems when dealing with high dimensional data.