Report On Nearest Neighbors

Student Details:

Student Name and ID of the member submitting the assignment:

Shivani Manojkumar Panchiwala - 1001982478

Student Name and ID of the remaining members:

Kuldip Rameshbhai Savaliya - 1001832000 &

Meghaben Ghanshyambhai Patel – 1002006777

Introduction of the Nearest Neighbors method

K Nearest Neighbor (KNN) method is intuitive to understand, an easy to implement and executes quickly, but also has a "greedy" nature. Nearest neighbor classification is a machine learning method that aims at labelling previously unseen query objects while distinguishing two or more destination classes. As any classifier, in general, it requires some training data with given labels and, thus, is an instance of supervised learning.

Here, nearest neighbors are those data points that have minimum distance in feature space from our new data point. And K is the number of such data points we consider in our implementation of the algorithm. Therefore, distance metric and K value are two important considerations while using the KNN algorithm.

Minkowski distance is the most popular distance metric among many others such as Euclidean distance, hamming distance, Manhattan distance, which can be used as per your need.

As the name suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new Datapoint. K Nearest Neighbor method falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets.

Despite being the laziest among algorithms, KNN has built an impressive reputation and is a go-to algorithm for several classification and regression problems. Of course, due to its laziness, it might not be the best choice for cases involving large data sets. But it's one of the oldest, simplest, and accurate algorithms out there.

Explain all the pre-processing data in detail And Explain all the parameters of KNN in details in your own words

Packages Requirements:

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt
```

Read and Load the Dataset

```
df = pd.read_csv("pima-indians-diabetes.csv") # Read the CSV File
df.head() # Get first 5 rows of dataframe
#https://www.shanelynn.ie/python-pandas-read-csv-load-data-from-csv-files/
```

	Preg	Plas	Pres	skin	test	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Here, df is a variable. pd is pandas library and read_CSV is used for to load or read the csv file.

```
# Finding missing values of the columns
df.isnull().sum()
#https://note.nkmk.me/en/python-pandas-nan-judge-count/
```

```
Preg     0
Plas     0
Pres     0
skin     0
test     0
mass     0
pedi     0
age      0
class    0
dtype: int64
```

Isnull is a function for finding missing value and sum() is used for finding sum.

```
df.info()
```

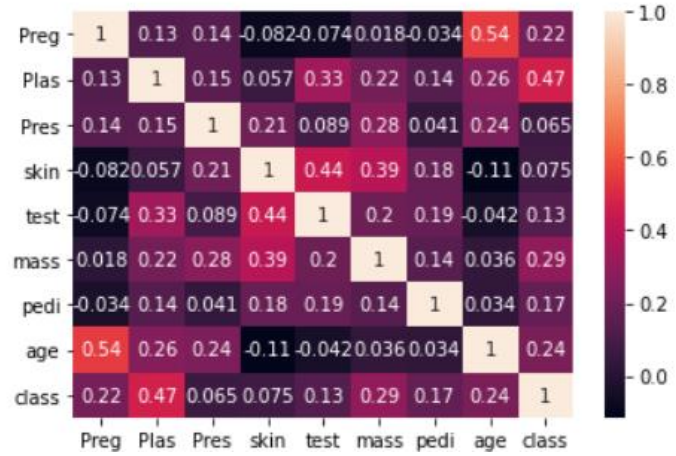
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Preg    768 non-null    int64  
 1   Plas    768 non-null    int64  
 2   Pres    768 non-null    int64  
 3   skin    768 non-null    int64  
 4   test    768 non-null    int64  
 5   mass    768 non-null    float64
 6   pedi    768 non-null    float64
 7   age     768 non-null    int64  
 8   class   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
df.describe()
```

	Preg	Plas	Pres	skin	test	mass	pedi	age	class
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
# Correlation Matrix for finding best attributes

sns.heatmap(df.corr(),annot=True)
plt.show()
#https://likegeeks.com/python-correlation-matrix/
```



sns is used for import seaborn library. And heatmap is one type of visualization to show. Where .corr() is used for correlation method.

```
## The target variable is 'class'

X = df[['Preg','Plas','Pres','skin','test','mass','pedi','age']]

y = df['class']

#Note: 0 – Non Diabetic Patient and 1 – Diabetic Patient
```

I took X variable for independent value and y is a target value.

```
# Univariate Selection for finding best attributes
# UNIVARIATE FEATURE SELECTION

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
# Apply SelectKBest class to rate attributes
score_func = SelectKBest(score_func=chi2, k='all').fit(X,y)
df_score = pd.DataFrame(score_func.scores_)
df_column = pd.DataFrame(X.columns)
# Concatenating two df_score and df_column to view the scores of corresponding columns
feature_score = pd.concat([df_column,df_score],axis=1)
feature_score.columns = ['Attributes','Score']
# Print 7 best features
print(feature_score.nlargest(7, 'Score'))
#https://medium.com/@nmscott14/3-feature-selection-methods-e7ccd6dbf316
```

```
  Attributes        Score
4       test  2175.565273
1       Plas  1411.887041
7        age   181.303689
5       mass   127.669343
0       Preg   111.519691
3       skin    53.108040
2       Pres    17.605373
```

For finding the best K value, I used univariate selection method. I import some library from sklearn. After that I apply select K Best class to rate attributes and fit X and y. after that concate score and columns to view the score of corresponding value. And then print 7 best features.

```
# Drop other columns and keep 3 best features columns

attr_drop2 = ['mass', 'Preg', 'skin', 'Pres', 'pedi']
df = df.drop(attr_drop2, axis=1)
X = X.drop(attr_drop2, axis=1)
print(X)
print(y)
#https://stackoverflow.com/questions/13411544/delete-a-colu
```

```
     Plas  test  age
0     148     0   50
1      85     0   31
2     183     0   32
3      89    94   21
4     137   168   33
..    ...   ...  ...
763   101   180   63
764   122     0   27
765   121   112   30
766   126     0   47
767    93     0   23

[768 rows x 3 columns]
0      1
1      0
2      1
3      0
4      1
      ..
```

After selecting top 3 best attribute, I drop other columns which is not required and print X and y value.

```
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=2022)
#print(X_train)
X_train, X_val, y_train, y_val = train_test_split(
    X_train, y_train, test_size=0.25, random_state=2022)
```

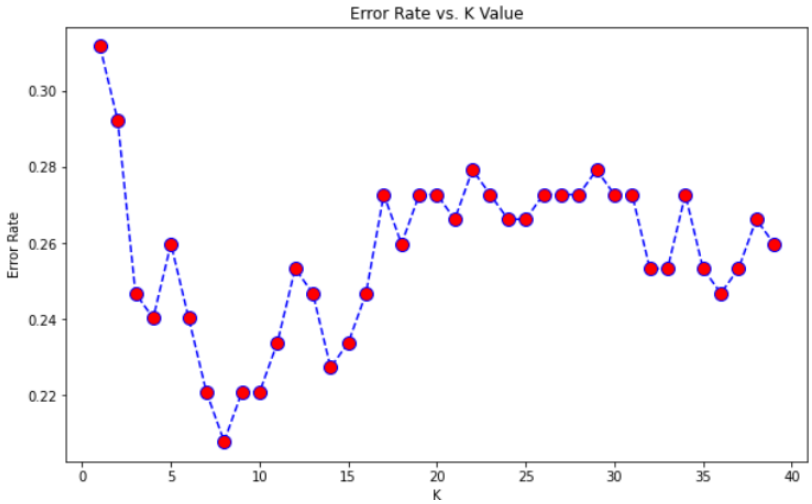
I used library train_test_split. I split dataset into 60% training set and 20% into testing set and set as 2022 random state. After that I split into 20% validation set on 2022 random state.

```
# Finding the Best K with minimum Error Rate
from sklearn.neighbors import KNeighborsClassifier
error_rate = []
# Might take some time
for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

For finding the best K value in another way, I used elbow method. I import KNeighborsclassifier. And give the range between 1 to 40 and predict the value. After that I plot the figsize.



From the graph, the value of K starts increasing from K = 8, which can be designated as the e lbow. Hence, K = 8 is the best K-value

To test the classifier, I used Minkowski distance with three different K value which is 8, 22, 35 Result with confusion matrix and classification Report.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report,confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report

knn1 = KNeighborsClassifier(n_neighbors = 8, metric = 'minkowski')
knn1.fit(X_train, y_train)
y_pred1 = knn1.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred1))
print(confusion_matrix(y_test, y_pred1))
print("Classification Report for K =8\n", classification_report(y_test, y_pre
#https://www.ritchieng.com/machine-learning-k-nearest-neighbors-knn/
```

Interpret and compare the results explain in details

0.7922077922077922
[[104 4]
[28 18]]
Classification Report for K =8

	precision	recall	f1-score	support
0	0.79	0.96	0.87	108
1	0.82	0.39	0.53	46
accuracy			0.79	154
macro avg	0.80	0.68	0.70	154
weighted avg	0.80	0.79	0.77	154

0.7207792207792207
[[97 11]
[32 14]]
Classification Report for K =22

	precision	recall	f1-score	support
0	0.75	0.90	0.82	108
1	0.56	0.30	0.39	46
accuracy			0.72	154
macro avg	0.66	0.60	0.61	154
weighted avg	0.69	0.72	0.69	154

0.7467532467532467
[[95 13]
[26 20]]
Classification Report for K =35

	precision	recall	f1-score	support
0	0.79	0.88	0.83	108
1	0.61	0.43	0.51	46
accuracy			0.75	154
macro avg	0.70	0.66	0.67	154
weighted avg	0.73	0.75	0.73	154

Classificaton For Validation

0.7077922077922078
[[77 11]
[34 32]]
Classification Report for K =8

	precision	recall	f1-score	support

0	0.69	0.88	0.77	88
1	0.74	0.48	0.59	66
accuracy			0.71	154
macro avg	0.72	0.68	0.68	154
weighted avg	0.72	0.71	0.69	154

0.7532467532467533
[[83 5]
[33 33]]
Classification Report for K =22

	precision	recall	f1-score	support
0	0.72	0.94	0.81	88
1	0.87	0.50	0.63	66
accuracy			0.75	154
macro avg	0.79	0.72	0.72	154
weighted avg	0.78	0.75	0.74	154

0.7337662337662337
[[79 9]
[32 34]]
Classification Report for K =35

	precision	recall	f1-score	support
0	0.71	0.90	0.79	88
1	0.79	0.52	0.62	66
accuracy			0.73	154
macro avg	0.75	0.71	0.71	154
weighted avg	0.75	0.73	0.72	154

Show the best one

Using minkowski distance, I choose 3 different k-values which is 8, 22, 35 and fit for the training and testing and validation.

Based on the Accuracy, n_neighbors = 8 has 0.7922077922077922 more accuracy which is higher than other.

Explain what your criteria was for selecting the three attributes.

What other 3 attribute can you choose?

We used Correlation matrix and Univariate feature selection (Select K-best) for selecting the best 3 attributes. ▪ In Correlation matrix, 0 indicates little to no linear relationship. +1

indicates a perfect positive linear relationship: as one variable increases in its values, the other variable also increases in its values via an exact linear rule. -1 indicates a perfect negative linear relationship: as one variable increases in its values, the other variable decreases in its values via an exact linear rule. Values between 0 and 0.3 (0 and -0.3) indicate a weak positive (negative) linear relationship via a shaky linear rule. Values between 0.3 and 0.7 (-0.3 and -0.7) indicate a moderate positive (negative) linear relationship via a fuzzy-firm linear rule. Values between 0.7 and 1.0 (-0.7 and -1.0) indicate a strong positive (negative) linear relationship via a firm linear rule. Features with high correlation are more linearly dependent and hence have almost the identical effect on the dependent variable.

In Select K-best features, A universal function Select K Best which can select k-best features based on some metric, you only need to provide a score function to define your metric. Luckily, sklearn provides some predefined score functions. We used Chi-squared(chi2) stats of non-negative features for classification tasks.

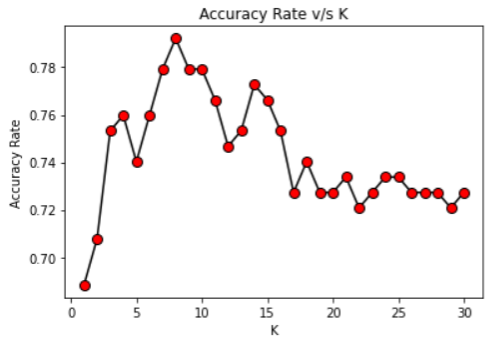
By observing the relationship values, if we must consider other 3 attributes than they will be ‘Age’, ’Plas’ and ‘test’.

Plot the classifier in 2D Projection and my observation

```
accuracy_rate=np.zeros(30)
for n in range(1,31):
    knn_acc=KNeighborsClassifier(n_neighbors=n)
    knn_acc.fit(X_train,y_train)
    pred_acc = knn_acc.predict(X_test)
    accuracy_rate[n-1]=metrics.accuracy_score(y_test, pred_acc)

plt.plot(range(1,31),accuracy_rate,color='black', marker='o', markerfacecolor
plt.title('Accuracy Rate v/s K')
plt.xlabel('K')
plt.ylabel('Accuracy Rate')
plt.show()
print(accuracy_rate)
```

In this plotting K value against respective accuracy score obtained from classifier corresponding to k = 8.




```
[0.68831169 0.70779221 0.75324675 0.75974026 0.74025974 0.75974026
 0.77922078 0.79220779 0.77922078 0.77922078 0.76623377 0.74675325
 0.75324675 0.77272727 0.76623377 0.75324675 0.72727273 0.74025974
 0.72727273 0.72727273 0.73376623 0.72077922 0.72727273 0.73376623
 0.73376623 0.72727273 0.72727273 0.72727273 0.72077922 0.72727273]
```

This graph visualizes the accuracy score obtained from the classifier against different values of K from 1 to 30. We can see that K = 8 gives the highest accuracy as also seen from the Error rate v/s K graph.

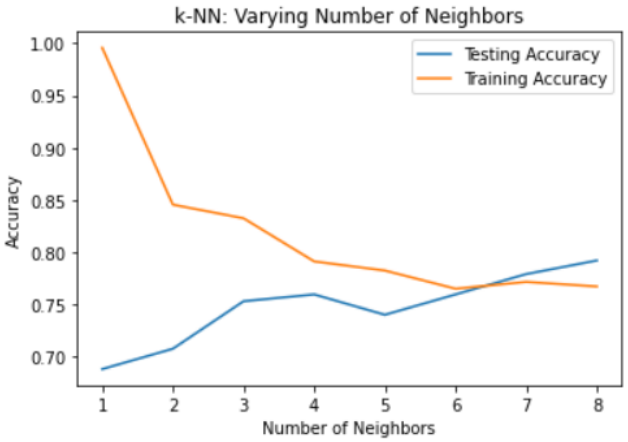
```
# K value VS Accuracy
no_neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(no_neighbors))
test_accuracy = np.empty(len(no_neighbors))

for i, k in enumerate(no_neighbors):
    # We instantiate the classifier
    knn = KNeighborsClassifier(n_neighbors=k)
    # Fit the classifier to the training data
    knn.fit(X_train,y_train)

    # Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)

    # Compute accuracy on the testing set
    test_accuracy[i] = knn.score(X_test, y_test)

plt.title('k-NN: Varying Number of Neighbors')
plt.plot(no_neighbors, test_accuracy, label = 'Testing Accuracy')
plt.plot(no_neighbors, train_accuracy, label = 'Training Accuracy')
plt.legend()
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.show()
```

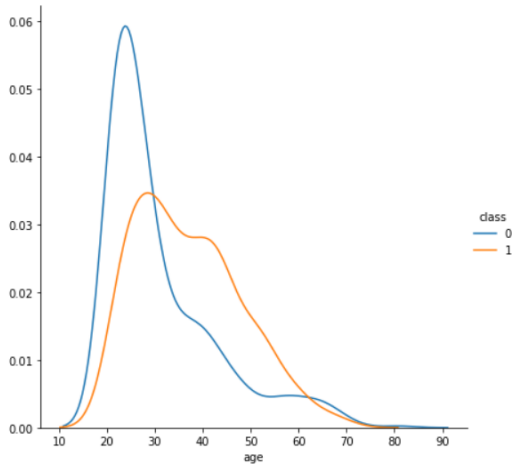


In this plot we can see that training accuracy is higher than testing accuracy.

```
sns.FacetGrid(df, hue="class", size=6) \
   .map(sns.kdeplot, "age") \
   .add_legend()
```

Distribution density plot KDE (kernel density estimate) between age and class

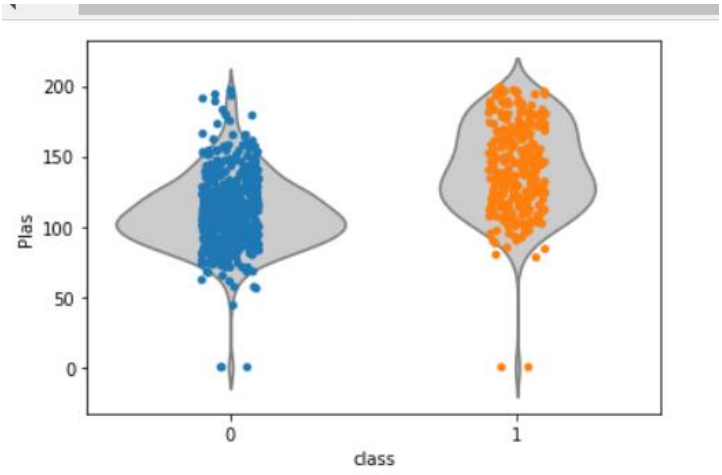
#Plotting bivariate relations between each pair of features (4 features x4 so 16 graphs) with hue = "class"



This Distribution density plot portrays the mortality and class count for each age(0 – Non Diabetic Patient and 1 — Diabetic Patient). We can see that, between age of 10 to 70 are less diabetic patient.

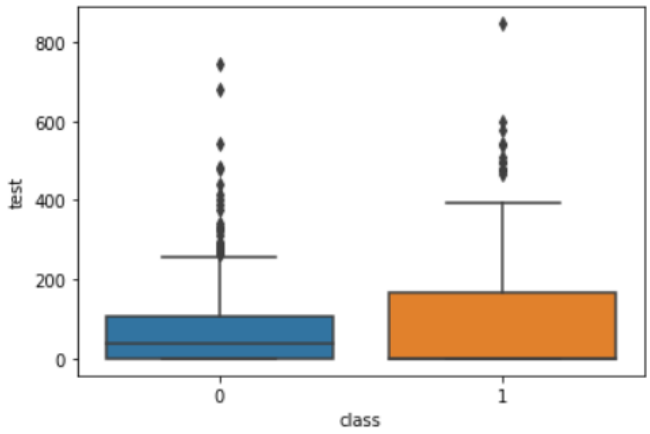
```
ax = sns.violinplot(x="class", y="Plas", data=df,
                    inner=None, color=".8")
ax = sns.stripplot(x="class", y="Plas", data=df)
```

Scatter and Violin Plot to visualise the class Rate density with respect to Plas




```
sns.boxplot(x="class", y="test", data=df)
```

Boxplot to visualise the class Rate density with respect to test



References:

https://www.shanelynn.ie/python-pandas-read-csv-load-data-from-csv-files/

https://note.nkmk.me/en/python-pandas-nan-judge-count/

https://likegeeks.com/python-correlation-matrix/

https://medium.com/@nmscott14/3-feature-selection-methods-e7ccd6dbf316

https://stackoverflow.com/questions/13411544/delete-a-column-from-a-pandas-dataframe

https://datascience.stackexchange.com/questions/15135/train-test-validation-set-splitting-in-sklearn

https://datascienceplus.com/k-nearest-neighbors-knn-with-python/

https://datascienceplus.com/k-nearest-neighbors-knn-with-python/

https://www.ritchieng.com/machine-learning-k-nearest-neighbors-knn/

https://datascienceplus.com/k-nearest-neighbors-knn-with-python/

https://www.w3resource.com/machine-learning/scikit-learn/iris/python-machine-learning-k-nearest-neighbors-algorithm-exercise-8.php

https://www.kaggle.com/code/grim1412/seaborn-visualization-and-knn-on-the-iris-dataset/notebook

https://seaborn.pydata.org/generated/seaborn.stripplot.html

https://seaborn.pydata.org/generated/seaborn.violinplot.html

https://www.kaggle.com/code/grim1412/seaborn-visualization-and-knn-on-the-iris-dataset/notebook