K-Nearest Neighbors (K-NN)

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

```
dataset =  pd.read_csv('Class.csv')
```

Data Analysis EDA

```
dataset.shape
```

```
(1000, 5)
```

```
dataset.head()
```

```
   Feature1  Feature2  Feature3  Feature4 Class
0  0.012639 -1.143888 -1.779450  0.680949     B
1  0.038752 -1.750615 -1.125835  3.552359     B
2  0.680677 -1.528170 -1.913719  0.810822     B
3 -0.224386  0.139624 -1.257102  3.134959     B
4 -0.489983  0.183918  1.727542 -1.696429     A
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Feature1  1000 non-null   float64
 1   Feature2  1000 non-null   float64
 2   Feature3  1000 non-null   float64
 3   Feature4  1000 non-null   float64
 4   Class     1000 non-null   object
dtypes: float64(4), object(1)
memory usage: 39.2+ KB
```

```
dataset.describe()
```

```
          Feature1     Feature2     Feature3     Feature4
count  1000.000000  1000.000000  1000.000000  1000.000000
mean     -0.008864     0.007520    -0.480925    -0.481364
std       1.019564     0.958155     1.537991     1.650390
min      -3.278297    -3.310776    -4.325728    -4.790634
25%      -0.715176    -0.616532    -1.639231    -1.595664
```

```
50%      -0.052028    0.003680   -1.090309   -1.243851
75%       0.703675    0.678094    1.286118    0.834556
max       3.734298    2.660162    2.578732    4.574713
```

```
dataset.groupby('Class').size()
```

```
Class
A    337
B    333
C    330
dtype: int64
```

Data Preprocessing

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.20,
                                                    random_state = 0)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(X_train)

X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

```
print(X_train)
```

```
[[ 1.44452751 -0.04729699  1.58131018 -0.97570824]
 [-0.22843541  0.06112275 -1.50904153 -1.87783849]
 [ 0.37673734  0.06932052 -0.62501277  1.71445959]
 ...
 [ 0.39496783  1.33316344  1.05713686  0.59628126]
 [-0.89844375 -0.48942822  1.5483909  -0.74791369]
 [-0.55815164 -1.0263108   1.36406086 -0.63952184]]
```

```
print(X_test)
```

```
[[-0.91528992  0.09227092  0.29430048  0.31954243]
 [-0.61576181  1.6565001  -1.36808122 -1.30586151]
 [ 0.15734056  0.07818213  1.69833949 -0.86266571]
 [-1.10109056  0.36346088 -0.38159764  1.56598291]
 [-0.05543627 -0.18581803 -0.84155323 -0.37567232]
```

```
 [ 0.03700575 -0.14680175  1.26994629 -0.59822221]
 [ 0.64801277 -0.21259941 -1.16485579 -1.13171951]
 [-0.49081986 -3.02955585 -0.88990286 -0.93616582]
 [ 2.01505227  0.38456685 -0.78702335  0.86173111]
 [ 0.08770621 -0.13417387 -0.67283484  0.17641288]
 [-1.00547376  0.02786478 -0.58172747 -0.54330928]
 [ 0.39687971 -1.06137449 -0.851849    0.53980399]
 [ 0.96191645 -1.17954138 -0.80689013 -1.26102695]
 [ 1.36951165 -0.4932265   1.12567721 -0.61722122]
 [ 0.13060585  0.45712857 -0.98258359  0.83546483]
 [ 1.61722048  2.33836092 -0.78506763 -1.40516666]
 [-0.92768281 -0.57847332 -1.29170106 -0.44030367]
 [-0.70175706 -0.42962133  1.4880354  -0.79396162]
 [-1.18467125  0.68988885  0.8384472  -0.51334454]
 [ 1.59348745 -0.04669995  1.57439209 -1.01929082]
 [ 1.44414579  0.02713829 -0.06308767  0.18451396]
 [ 1.31643276 -0.64091058 -0.7092261   0.90592815]
 [-0.02870832 -0.21989748 -0.83103085  0.71753359]
 [ 0.15912071  1.34608092 -0.62711399  0.86938185]
 [ 0.28800318  2.01979875 -2.05472976 -1.73093286]
 [-0.71721057  0.71859794  1.06050045 -0.5581394 ]
 [ 1.09067162 -0.18153078 -0.64976884 -1.01742307]
 [-1.41854556  0.42040135 -0.64459452  0.48314975]
 [-1.05074883  1.25328635 -0.2771732   0.1008558 ]
 [ 0.06092773 -1.86172271 -0.4350034   2.47003348]
 [-0.53253983  0.47855954  1.34003144 -0.69066355]
 [ 0.75282865 -0.46331438  1.58063273 -0.84714973]
 [-0.71964019 -0.50707091 -0.50881098  1.1014666 ]
 [ 0.18050894 -1.00403885  1.30850555 -0.77447146]
 [ 1.62744677 -0.37784449 -0.73243937  1.25548345]
 [-1.04122112 -1.41980169 -0.78438544 -0.3059955 ]
 [-0.10634306 -0.79782561 -0.8198954   0.99263513]
 [-0.30782318  0.70224126 -1.1516624   1.0267794 ]
 [ 0.07982933 -0.60889111  1.16529597 -0.66723993]
 [ 1.19624835  0.36677328 -1.43809197 -1.37621921]
 [-0.75465274  1.17990184 -0.55680642 -0.39219574]
 [-0.19892487 -0.33120047 -0.99767276  0.86290451]
 [ 0.32300068 -0.46346467 -0.89358007  0.79714889]
 [ 1.41916684  0.14748651 -0.6842756   1.63408697]
 [ 0.2151897   0.34735191 -0.92502235  0.641022  ]
 [ 1.2706119   1.32490423 -0.25897618  2.109834  ]
 [ 1.69681876 -0.85914664  1.38310695 -0.77617329]
 [ 1.93121985  0.62958542 -0.59910082  0.93827991]
 [-0.31810305 -1.33629212  1.2945598  -0.79219784]
 [-2.11639258 -0.14272192 -0.67103849 -1.53986959]
 [ 0.43487345  0.3903654  -0.69382887  1.63004431]
 [-0.16360057 -1.30411359  0.35996319  0.1751273 ]
 [-0.63760389  1.17213986  1.22520025 -0.68008148]
 [-0.37604129 -0.77852632 -0.3660779  -0.57644488]
```

```
 [-0.1738982   0.14993422 -0.94180334 -0.39076135]
 [ 0.99191698  1.41533233 -0.76417884 -0.58169989]
 [-0.22080679  0.27589895 -0.86122445  1.66785692]
 [ 1.4176248  -0.72204991 -0.2980222  -0.5820174 ]
 [ 0.1539836   0.72710709 -0.07752022 -0.82674836]
 [ 1.34680678 -1.45205339  1.17196038 -0.52922466]
 [-0.23201928 -0.96567249  0.50428561  0.06488889]
 [-0.57254897  1.41817308 -1.11763885 -1.31296627]
 [ 1.12885926  1.00720596  1.3366727  -0.67332948]
 [ 1.42067836  0.42033752  0.04275873  0.05656196]
 [ 1.1357713   0.47500049 -0.42296112  2.06838231]
 [-1.09686957 -0.2749047  -0.75198438  0.97276454]
 [ 1.25805917  0.60705598 -0.7702204   1.10080723]
 [ 0.06605416 -0.18860809 -0.51309252  1.15175205]
 [-0.32472001  1.44044867  0.81567426 -0.293092  ]
 [ 1.33156213  1.83673727  0.59074899  0.38633459]
 [ 2.65853404  0.66299005  1.38562411 -0.8045592 ]
 [-0.88713564 -0.31845308 -0.69263423 -1.21270145]
 [-0.41004542  0.35703264 -0.23733641 -0.27175414]
 [-0.25372121 -1.65916976  0.75629535 -0.35711268]
 [-0.49270831  1.2783115  -0.70385865 -0.49825002]
 [-1.65797146  2.67066691  1.25073792 -0.61296828]
 [-0.87364292 -1.29369253 -1.03105457 -0.94769306]
 [ 1.27644114 -1.21671569 -0.01077195  0.06347735]
 [ 1.31102263  1.41148543 -1.19258797 -0.00486584]
 [ 0.18498752  0.28397977 -0.34215899  0.16110726]
 [-0.61464928  1.17165001 -0.72233016 -0.7627653 ]
 [-0.72641844 -0.41737779  1.36052934 -0.7179056 ]
 [-0.7093283   1.29850241 -0.37751635  1.32150649]
 [ 0.62892633 -2.11533802  1.36816868 -0.67022198]
 [-0.34624672 -0.80662421 -0.73346813  1.43111917]
 [-0.50036196 -0.79034882 -0.69228797  1.34894431]
 [-0.7837944   0.99153367  1.10420314 -0.71917119]
 [ 0.11173357 -0.95554107 -0.66361382 -0.85393759]
 [-0.79892333  0.52702513 -0.32934449  2.05568104]
 [-1.13869709 -0.10384246 -0.2652259  -0.48137333]
 [-0.30732269 -0.38404113  0.81050355 -0.32427364]
 [-1.61331925 -0.11918641 -0.70778431  1.62247224]
 [ 0.15026369 -1.15249378 -0.73324455 -0.56880359]
 [ 1.5586879   0.02952286 -0.311973   -0.25891273]
 [ 0.46871535 -1.53464457  1.40354002 -0.78689554]
 [-0.12840059 -0.60473826 -0.85195399  1.27620526]
 [-0.28528165 -1.97606969  1.27838578 -0.59951442]
 [ 1.13693861 -0.50467395  1.1886776  -0.70476167]
 [ 0.07264391 -0.17103292 -0.1623744  -0.45344201]
 [ 0.56273657 -1.68742476  1.12908029 -0.76725715]
 [ 0.06256838 -1.06176042  1.12765385 -0.67488856]
 [-1.68091825 -0.41684378 -0.83320234 -0.59546721]
 [ 0.2670546  -0.1768855   1.54932548 -0.62620609]
```

```
 [ 0.33199633  1.02209968 -1.5232681  -1.81200704]
 [-0.05630493  2.73676548 -0.37276764  1.59540821]
 [ 0.57685035  0.05940398 -0.43738501  1.32642429]
 [ 2.11442274 -0.16781337  0.38610706 -0.22329101]
 [ 0.23704298  1.17513506 -0.44885765  1.27026667]
 [ 0.5280946  -0.45221284 -0.99929655 -0.51366488]
 [ 0.57019378 -0.28644655  0.12086131  0.39342101]
 [ 1.31371908  0.08733375 -1.19301222 -1.41752929]
 [ 1.31940534 -0.51354569  1.16098797 -0.71674298]
 [-0.68176822 -0.34406813 -0.59169417  2.79553212]
 [-0.1693568   1.37995065  0.01358598  0.22105369]
 [-1.54373689 -0.36945447  1.28020116 -0.72570031]
 [-0.10894656 -0.35187278 -0.72735589  1.44165533]
 [ 0.12948824  0.04693377 -1.0853268  -1.54716214]
 [-0.9624083   0.54669714  1.08443958 -0.56108717]
 [ 0.51503859  2.04730259 -1.32406062 -1.63783016]
 [ 0.30342576 -0.3444814  -0.57587666 -0.6827299 ]
 [ 1.01549201 -0.97359888  1.61050398 -0.81294548]
 [ 1.68788326 -2.31053023 -0.59261735  2.56786737]
 [-0.50021734 -0.62925866 -0.56003804 -0.4421084 ]
 [ 0.26432053 -1.31978465 -2.31720661 -2.66363817]
 [ 0.37836509 -1.63800237 -0.41994772  3.03648879]
 [ 0.80872964 -0.28524688  1.11996465 -0.59651125]
 [-0.42099542  1.14925012 -0.46568127 -0.24281981]
 [ 1.85639036 -1.50879141 -0.09893022 -0.43189289]
 [ 1.68736678 -0.03274212 -0.15867891 -0.56238453]
 [-0.29735912 -0.93126953 -1.36002713 -1.06730027]
 [-1.00703615 -0.44916176  1.17265486  1.3752873 ]
 [ 0.13891467 -1.36351309 -0.76701562  0.67071097]
 [-0.56422558 -1.71395343  1.06459395 -0.61528157]
 [ 0.14661607 -1.52781176 -0.71562745  1.48911749]
 [ 1.13790636 -0.41188766 -0.98499693  0.62517534]
 [-0.09805891  1.17923798 -0.23767752 -0.1069328 ]
 [ 1.87831679 -0.84286398 -1.19996884 -0.43270672]
 [-0.85123059 -0.77111757 -0.28685749 -0.5383183 ]
 [ 0.32242838  0.34465891 -0.05230802 -0.1581072 ]
 [-0.47236328 -0.5294448  -0.77728517 -0.73412524]
 [ 0.24658686  0.82259502  1.39706494 -0.71660195]
 [ 0.03820751 -0.05806558  1.18086991 -0.5858678 ]
 [ 1.89761688  0.54348255 -0.1044256   2.59289228]
 [-1.22094083 -2.05557709 -1.64584481 -2.06283469]
 [-0.39629049 -0.33324939 -0.86010873  0.20634747]
 [-0.41234054  0.66687381  1.14139326 -0.67698163]
 [-2.10360736  1.31546077  0.01126091  0.02765417]
 [ 1.35702973 -0.55275727 -0.24472762  2.21149638]
 [-0.3521298   0.52488589  1.36977553 -0.76755177]
 [ 1.18911675  0.33988076 -0.46633017 -0.71641647]
 [ 0.15246235 -0.27101279  1.14039789 -0.53968295]
 [-1.43880549  1.04997242 -0.54154486 -0.80110482]
```

```
 [ 1.12708265 -0.64624641 -0.97977362 -0.54981312]
 [-0.71133631  2.75870652  0.87871719 -0.5779471 ]
 [-1.63241137 -0.68772577 -0.55509709  1.98956485]
 [ 0.26753815 -0.29886962  0.3231322   0.41011084]
 [-0.16354425  1.14894374 -1.10619362 -0.81414272]
 [-0.49576148  0.93647541 -1.82024567 -1.35598596]
 [ 0.44328181 -1.14278247 -0.7708442   1.34130603]
 [ 0.2846987  -1.21588995 -0.81774097  1.35086202]
 [-1.13355712 -0.63930443 -0.87693509  0.58030564]
 [-1.09519592 -0.67083041  1.5411114  -0.83267348]
 [ 0.04940588 -0.41426542  1.03610193 -0.58131703]
 [ 1.70854784  0.07847541 -1.24327619 -0.24994696]
 [ 1.08875556 -1.50820781 -0.67186663 -0.80497262]
 [ 1.41829    -0.78333418  1.54038751 -0.73997979]
 [-0.29909275 -1.07725589 -0.39433142 -0.48103859]
 [ 0.75125879  0.31670253  1.35523782 -0.68839084]
 [-0.78482254  0.05381572  1.48927923 -0.67663811]
 [-0.5039221  -0.15652189 -0.42210592 -0.31489448]
 [ 0.08861987 -0.55659523 -1.25005071 -1.25150871]
 [ 0.26922373  0.00686944  0.11084075  0.76645003]
 [ 0.89267106  0.11372181 -0.47466327  1.38204354]
 [ 0.89596577 -0.93579302  1.35396311 -0.60400852]
 [-0.01229172 -0.05484124 -0.82493309  1.05618027]
 [-1.03156721  1.04417878 -1.13176363 -0.93236271]
 [ 1.42857836  0.02108542 -0.41051496  2.46248265]
 [-0.41329308  0.27074352  1.31635256 -0.60280689]
 [-0.58682724 -0.73605601 -1.82541144 -1.55044641]
 [-0.65662248 -1.23632599 -1.48712948 -1.54664098]
 [ 1.15782709 -1.38077858  1.46000958 -0.77376918]
 [-1.60367464 -1.47044591 -0.40880158  1.22404487]
 [-0.04306836  1.14525366  1.15332009 -0.58896511]
 [ 0.34093448 -1.28750802 -0.43410926  2.73178353]
 [-0.62539383  0.76727363 -0.69419135  1.86515831]
 [ 0.85523644 -0.0775068   1.33439188 -0.67851356]
 [-0.8803794  -0.10081847  1.38261717 -0.8036696 ]
 [-0.63461975  0.2925913  -1.07354673  0.78724283]
 [-1.05071915 -0.2854863   0.63979598 -0.43081473]
 [ 0.46431626  1.32903114  1.51889882 -0.50416069]
 [-0.66921276 -0.78287962 -0.39832064  2.0343704 ]
 [-0.62706823 -0.7642684  -0.50933604 -0.70707092]
 [-0.90689497  0.06288383  1.10330343 -0.6551085 ]
 [-1.31227321 -1.01881498 -1.04363206 -0.60207821]
 [-0.3252578  -1.8445946  -1.13521235 -0.78593333]
 [-0.15385344  0.2940559   1.3231015  -0.6254021 ]
 [-0.55736081 -0.94289258  1.54997098 -0.76963596]
 [ 0.29309562  2.58079295  1.63943463 -0.78823084]
 [-0.71613037  1.0167015  -0.54776842 -0.58391178]
 [-0.81147364 -0.44218551 -0.26664297 -0.24325816]]
```

Training the K-NN model on the Training set

```
from sklearn.neighbors import KNeighborsClassifier
classifier =
KNeighborsClassifier(n_neighbors=11,p=2,metric='cityblock')
classifier.fit(X_train,y_train)
```

```
KNeighborsClassifier(metric='cityblock', n_neighbors=11)
```

Getting nearest neighbours for each point in training data

```
classifier.kneighbors(X=X_train, n_neighbors=7, return_distance=False)
```

```
array([[  0, 424, 679, ..., 325, 385, 410],
       [  1,  43, 142, ..., 763, 619, 113],
       [  2, 483, 394, ..., 344, 638, 137],
       ...,
       [797, 191, 646, ..., 194, 116, 197],
       [798, 124, 717, ..., 228,  73, 109],
       [799, 121, 339, ..., 613,  82,  59]], dtype=int64)
```

```
dataset.iloc[[  0,  16,  73,  55,  54,  60,  29],-1]
```

```
0     B
16    B
73    A
55    C
54    A
60    B
29    C
Name: Class, dtype: object
```

```
classifier.predict(X_train[[1]])
```

```
array(['C'], dtype=object)
```

Predicting the Test set results

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len
(y_test),1)),1))
```

```
[['C' 'C']
 ['C' 'C']
 ['A' 'A']
 ['B' 'B']
 ['C' 'C']
 ['A' 'A']
 ['C' 'C']
 ['C' 'C']
 ['B' 'B']
```

```
['B' 'B']
['C' 'C']
['B' 'B']
['C' 'C']
['A' 'A']
['B' 'B']
['C' 'C']
['C' 'B']
['A' 'A']
['A' 'A']
['A' 'A']
['C' 'C']
['B' 'B']
['B' 'B']
['B' 'B']
['C' 'C']
['A' 'A']
['C' 'C']
['B' 'B']
['C' 'C']
['B' 'B']
['A' 'A']
['A' 'A']
['B' 'B']
['A' 'A']
['B' 'B']
['C' 'B']
['B' 'B']
['B' 'B']
['A' 'A']
['C' 'C']
['C' 'C']
['B' 'B']
['B' 'B']
['B' 'B']
['B' 'B']
['B' 'B']
['A' 'A']
['B' 'B']
['A' 'A']
['C' 'C']
['B' 'B']
['C' 'C']
['A' 'A']
['C' 'C']
['C' 'C']
['C' 'C']
['B' 'B']
['C' 'C']
```

```
['C' 'C']
['A' 'A']
['C' 'C']
['C' 'C']
['A' 'A']
['C' 'C']
['B' 'B']
['B' 'B']
['B' 'B']
['B' 'B']
['A' 'A']
['A' 'C']
['A' 'A']
['C' 'C']
['C' 'C']
['A' 'A']
['C' 'C']
['A' 'A']
['C' 'C']
['C' 'C']
['C' 'B']
['C' 'C']
['C' 'C']
['A' 'A']
['B' 'B']
['A' 'A']
['B' 'B']
['B' 'B']
['A' 'A']
['C' 'C']
['B' 'B']
['C' 'C']
['A' 'A']
['B' 'B']
['C' 'C']
['C' 'C']
['A' 'A']
['B' 'B']
['A' 'A']
['A' 'A']
['C' 'C']
['A' 'A']
['A' 'A']
['C' 'C']
['A' 'A']
['C' 'C']
['B' 'B']
['B' 'B']
['A' 'C']
```

```
 ['B' 'B']
 ['C' 'C']
 ['C' 'C']
 ['C' 'C']
 ['A' 'A']
 ['B' 'B']
 ['C' 'C']
 ['A' 'A']
 ['B' 'A']
 ['C' 'C']
 ['A' 'A']
 ['C' 'C']
 ['C' 'C']
 ['A' 'A']
 ['B' 'B']
 ['C' 'C']
 ['C' 'C']
 ['B' 'B']
 ['A' 'A']
 ['C' 'C']
 ['C' 'C']
 ['C' 'C']
 ['C' 'C']
 ['A' 'C']
 ['B' 'B']
 ['A' 'A']
 ['B' 'B']
 ['B' 'B']
 ['C' 'C']
 ['C' 'C']
 ['C' 'C']
 ['C' 'C']
 ['C' 'C']
 ['A' 'A']
 ['A' 'A']
 ['B' 'B']
 ['C' 'C']
 ['B' 'B']
 ['A' 'A']
 ['C' 'C']
 ['B' 'B']
 ['A' 'A']
 ['C' 'C']
 ['A' 'A']
 ['C' 'C']
 ['C' 'C']
 ['A' 'A']
 ['B' 'B']
 ['C' 'C']
```

```
 ['C' 'C']
 ['C' 'C']
 ['B' 'B']
 ['B' 'B']
 ['B' 'B']
 ['A' 'A']
 ['A' 'A']
 ['C' 'B']
 ['C' 'C']
 ['A' 'A']
 ['C' 'C']
 ['A' 'A']
 ['A' 'A']
 ['C' 'C']
 ['C' 'C']
 ['B' 'C']
 ['B' 'A']
 ['A' 'A']
 ['B' 'B']
 ['C' 'C']
 ['B' 'B']
 ['A' 'A']
 ['C' 'C']
 ['C' 'C']
 ['A' 'A']
 ['B' 'B']
 ['A' 'A']
 ['B' 'B']
 ['B' 'B']
 ['A' 'A']
 ['A' 'A']
 ['B' 'B']
 ['A' 'A']
 ['A' 'A']
 ['B' 'B']
 ['C' 'C']
 ['A' 'A']
 ['C' 'C']
 ['C' 'C']
 ['A' 'A']
 ['A' 'A']
 ['A' 'A']
 ['C' 'C']
 ['C' 'C']]
```

Evaluating the Algorithm

Making the Confusion Matrix & Predicting Accuracy Score

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test,y_pred)
print(cm)
accuracy = accuracy_score(y_test, y_pred)*100
print('Accuracy of our model is equal ' + str(round(accuracy, 2)) + '
%.')
```

```
[[59  2  0]
 [ 0 55  4]
 [ 3  1 76]]
Accuracy of our model is equal 95.0 %.
```

Making Classification Report

```
from sklearn.metrics import classification_report
# here f1 score is goodness of fit .
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           A       0.95      0.97      0.96        61
           B       0.95      0.93      0.94        59
           C       0.95      0.95      0.95        80

    accuracy                           0.95       200
   macro avg       0.95      0.95      0.95       200
weighted avg       0.95      0.95      0.95       200
```

Comparing Error Rate with the K Value

Parameter Tuning Using

```
from sklearn.model_selection import cross_val_score

# creating list of K for KNN
k_list = list(range(1,50))

# creating list of cv scores
cv_scores = []

# perform 10-fold cross validation
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=10,
```

```
scoring='accuracy')
    cv_scores.append(scores.mean())
```

plot the error values against K values

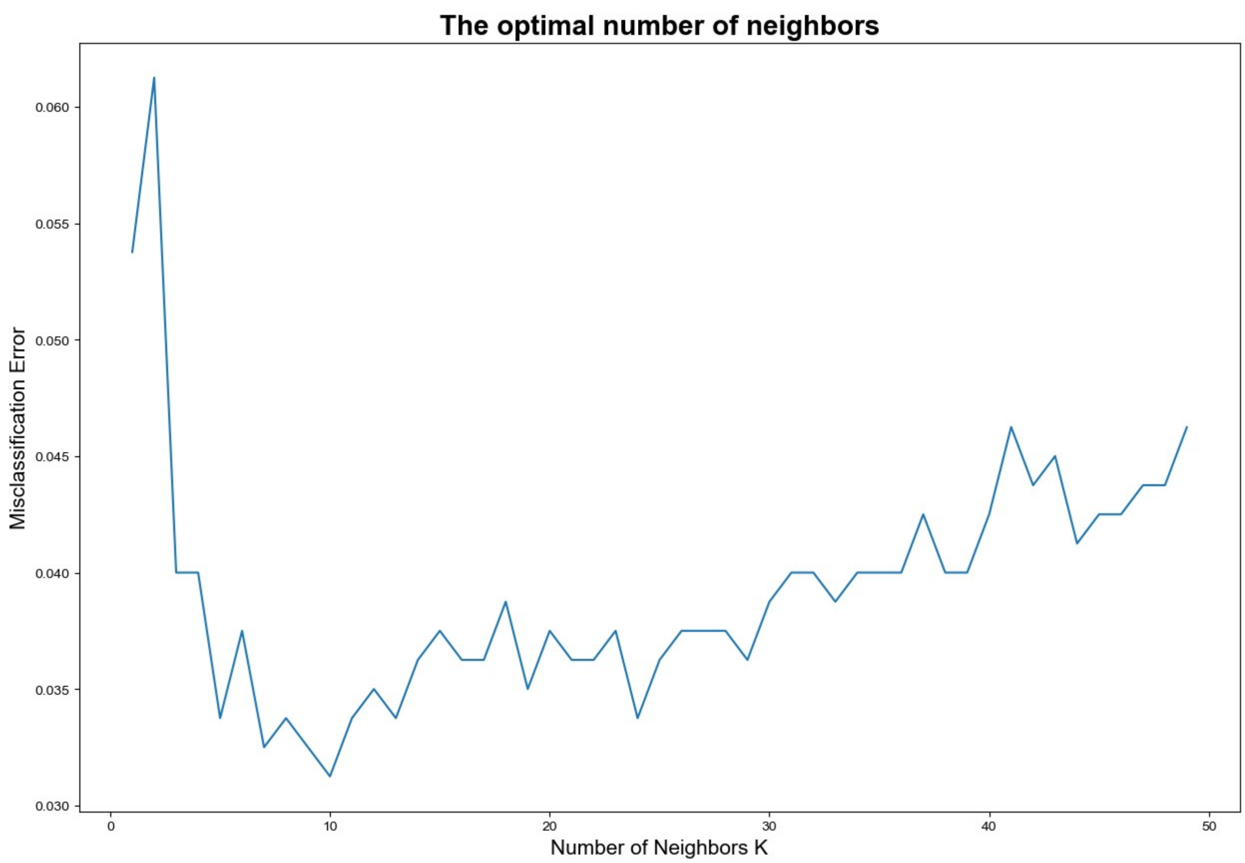
```
import seaborn as sns

# changing to misclassification error
MSE = [1-x for x in cv_scores]

plt.figure()
plt.figure(figsize=(15,10))
plt.title('The optimal number of neighbors', fontsize=20,
fontweight='bold')
plt.xlabel('Number of Neighbors K', fontsize=15)
plt.ylabel('Misclassification Error', fontsize=15)
sns.set_style("whitegrid")
plt.plot(k_list, MSE)

plt.show()

<Figure size 640x480 with 0 Axes>
```



finding best k

```
best_k = k_list[MSE.index(min(MSE))]
print("The optimal number of neighbors is %d." % best_k)
```

```
The optimal number of neighbors is 10.
```

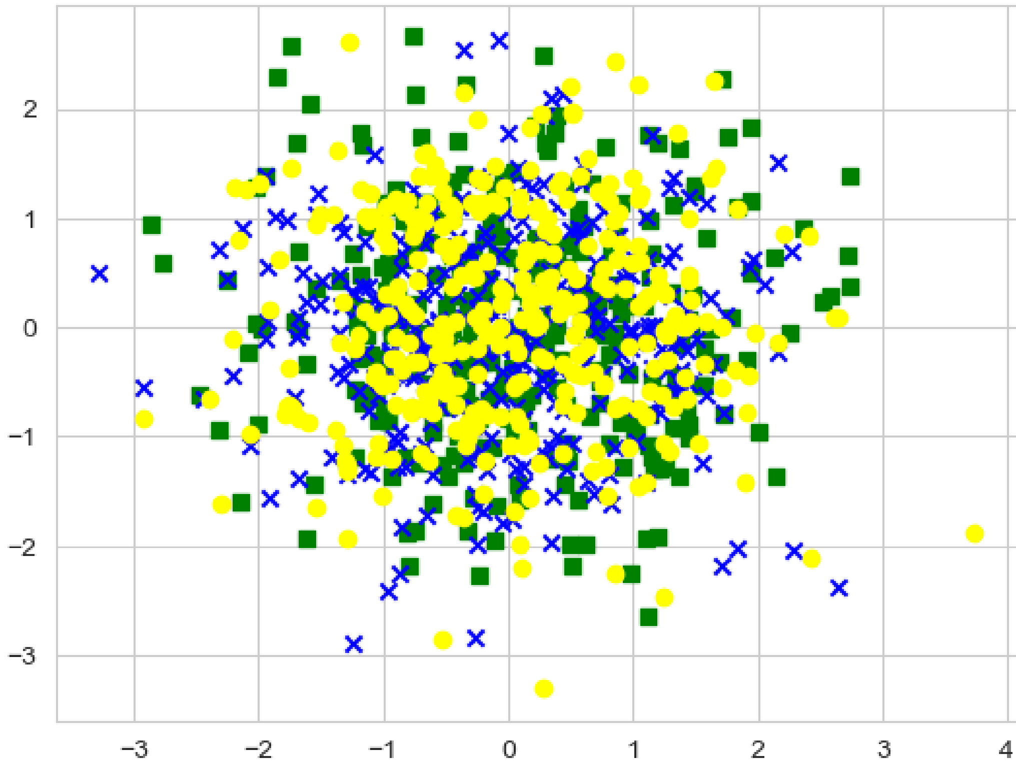
Visualize Test Result of KNN

```
from matplotlib.colors import ListedColormap

markers = ('s', 'x', 'o')
colors = ('green', 'blue', 'yellow')
cmap = ListedColormap(colors[:len(np.unique(y_test))])

for idx, cl in enumerate(np.unique(y)):
    plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],c=cmap(idx),
marker=markers[idx], label=cl)
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_39616\3368295777.py:8:
UserWarning: *c* argument looks like a single numeric RGB or RGBA
sequence, which should be avoided as value-mapping will have
precedence in case its length matches with *x* & *y*.  Please use the
*color* keyword-argument or provide a 2D array with a single row if
you intend to specify the same RGB or RGBA value for all points.
  plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],c=cmap(idx),
marker=markers[idx], label=cl)
```



2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
4