K-Nearest Neighbors Algorithm

In [4]:
```
# Import required libraries
import pandas as pd
import sklearn
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics.pairwise import euclidean_distances
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
```

In [6]:
```
student_data = pd.read_csv('student-mat.csv', sep=';')
data = student_data[['studytime', 'failures', 'absences', 'G1','G2','G3']]
data
```

Out[6]:

	studytime	failures	absences	G1	G2	G3
0	2	0	6	5	6	6
1	2	0	4	5	5	6
2	2	3	10	7	8	10
3	3	0	2	15	14	15
4	2	0	4	6	10	10
...	...	...	...	...	...	...
390	2	2	11	9	9	9
391	1	0	3	14	16	16
392	1	3	3	10	8	7
393	1	0	0	11	12	10
394	1	0	5	8	9	9

395 rows × 6 columns

In [7]:
```
# Display The names of all the columns in the data.
print(data.columns.values)
```

```
['studytime' 'failures' 'absences' 'G1' 'G2' 'G3']
```

In [10]:
```
# Choose features
x = data[['studytime', 'failures', 'absences' ,'G1' ,'G2']]
y = data[['G3']]
```

In [11]:
```
# Use train/test split with different random_state values
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

In [12]:
```
# Create a random forest classifier
# clf means classifier
clf = RandomForestClassifier(n_estimators=395, random_state=0, n_jobs=-1)

# Train the classifier
clf.fit(X_train, y_train.values.ravel())
```

Out[12]:
```
RandomForestClassifier(n_estimators=395, n_jobs=-1, random_state=0)
```

In [13]:
```
for feature in zip(x, clf.feature_importances_):
    print(feature)
```

```
('studytime', 0.11972146964546541)
('failures', 0.05493441817300572)
('absences', 0.24769763942172238)
('G1', 0.23720486524627107)
('G2', 0.34044160751353547)
```

In [14]:
```
# Check the number of testing examples
print(X_train.shape)
```

```
(296, 5)
```

In [15]:
```
# Classification (KNN)
# Create KNN Classifier
clf = KNeighborsClassifier(n_neighbors=23)
# Train the classifier
clf.fit(X_train, y_train.values.ravel())
```

Out[15]:
```
KNeighborsClassifier(n_neighbors=23)
```

In [16]:
```
# Test accuracy of KNN algorithm
a1 = accuracy_score(clf.predict(X_test), y_test)
print('Accuracy score for KNN algorithm =',a1*100,'%')
```

```
Accuracy score for KNN algorithm = 35.35353535353536 %
```

In [17]:
```
# Decision Tree Classification
# Create Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=0)
# Evaluate a score by cross-validation
cross_val_score(clf, X_train, y_train.values.ravel(), cv=10)
```

```
D:\Anaconda\lib\site-packages\sklearn\model_selection\_split.py:670: UserWarning: Th
e least populated class in y has only 1 members, which is less than n_splits=10.
  warnings.warn(("The least populated class in y has only %d"
```

Out[17]:
```
array([0.4       , 0.6       , 0.46666667, 0.36666667, 0.4       ,
       0.23333333, 0.37931034, 0.27586207, 0.31034483, 0.20689655])
```

In [18]:
```
# Train the classifier
clf.fit(X_train, y_train.values.ravel())
```

Out[18]:
```
DecisionTreeClassifier(random_state=0)
```

In [19]:
```
# Test accuracy of Decision Trees algorithm
a1 = accuracy_score(clf.predict(X_test), y_test)
print('Accuracy score for Decision Trees algorithm =',a1*100,'%')
```

```
Accuracy score for Decision Trees algorithm = 37.37373737373738 %
```

In [ ]: