الجامعة المستنصرية

كلية العلوم

قسم:الحاسوب

عنوان التقرير

Wheat seeds classification using nearest neighbor classification

أسم الطالب الرباعي:علي لؤي خلف جلوب

التــــــوقيع:

القسم: علوم الحاسبات **المـــرحلة:الثالثة**

الشعبــــة : A1 **الفــــــرع: CS**

البريد الالكتروني: luaya577@gmail.com

التقرير مقدم من ضمن متطلبات درجة الامتحان النهائي

لمادة ***أنظمة ذكية***

يملئ من قبل أستاذ المادة					
اسم الأستاذ:					
الدرجة					
من					

ختم اللجنة الامتحانية

توقيع المدقق

Abstract

K-nearest neighbors (KNN) K-Nearest Neighbors (KNN) is one of the simplest algorithms used in Machine Learning for regression and classification problem. KNN algorithms use data and classify new data points based on similarity measures (e.g. distance function). Classification is done by a majority vote to its neighbors.

K nearest neighbors (KNN) is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.[3]

1-Introduction

Hi everyone! Today I would like to talk about the K-Nearest Neighbors algorithm (or KNN). KNN algorithm is one of the simplest classification algorithm and it is one of the most used learning algorithms. So what is the KNN algorithm? I'm glad you asked! KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.Wheat seeds classification is an important agriculture process. In this paper a wheat classification system based on The nearest neighbor (KNN) .The proposed system aims to classify wheat seeds into their corresponding classes. The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. K-Nearest algorithm is one of the algorithm system used to solve classification problems. While algorithm is applied the matchings are done with the average of k-data appearing to be the closest as depended on the predetermined threshold value by comparing the similarity between data to be classified and normal behavior data in the learning set

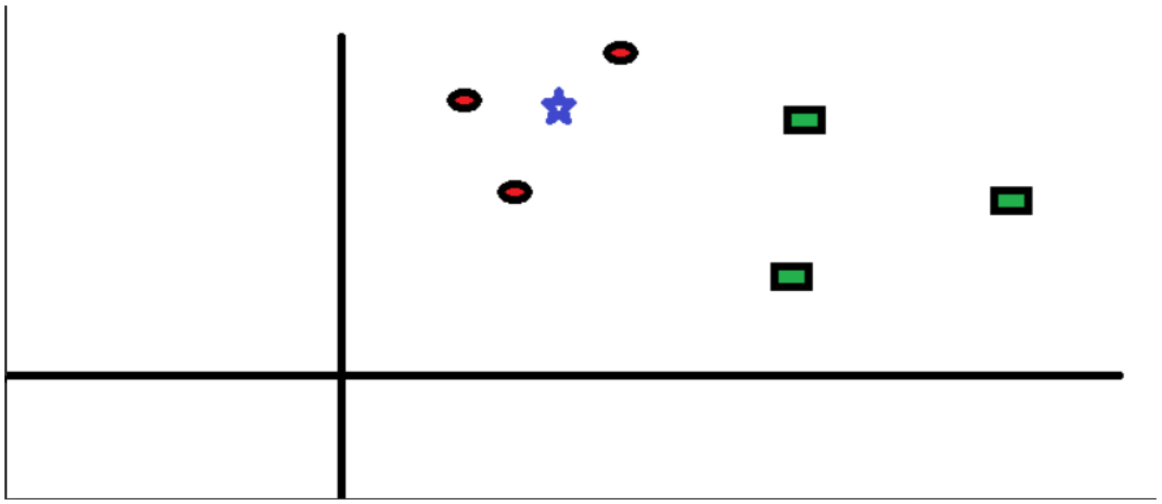
2-When do we use KNN algorithm?

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:-[2]

1. Ease to interpret output
2. Calculation time
3. Predictive Power

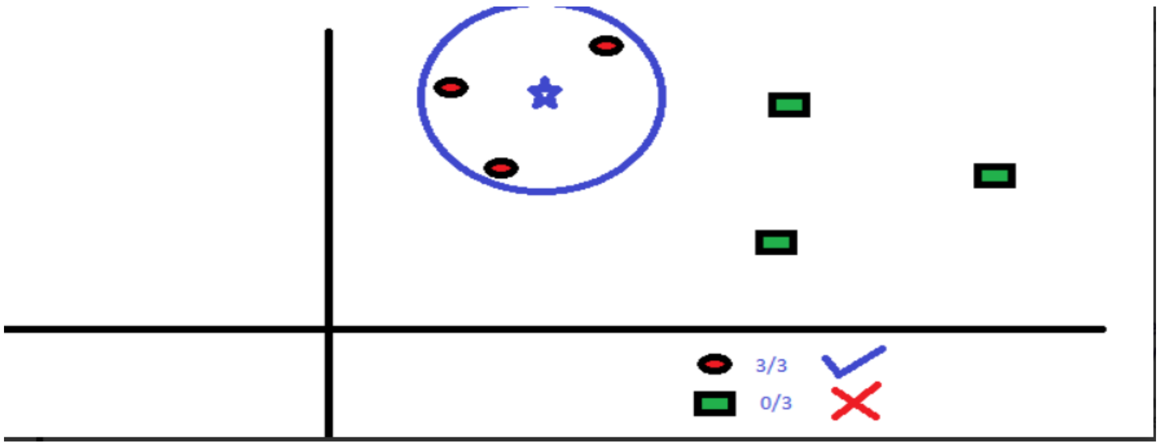
3-How does the KNN algorithm work?

Let's take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS). As shown in Figure 1



Figer1

You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else. The "K" is KNN algorithm is the nearest neighbor we wish to take the vote from. Let's say K = 3. Hence, we will now make a circle with BS as the center just as big as to enclose only three datapoints on the plane. Refer to the following diagram for more details: As shown in Figure 2 [2]

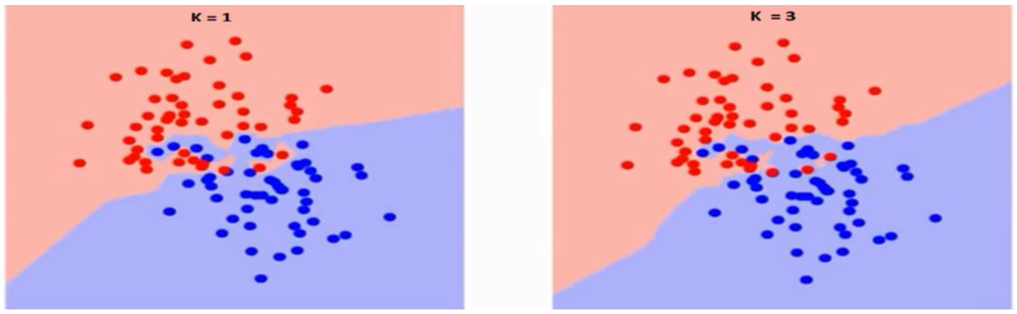


Figer2

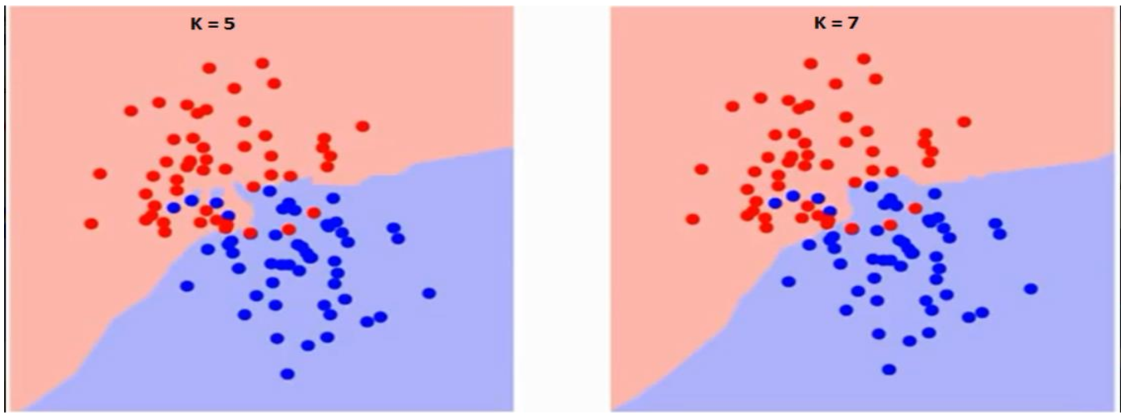
4-How do we choose the factor K?

First let us try to understand what exactly does K influence in the algorithm. If we see the last example, given that all the 6 training observation remain constant, with a given K value we can make boundaries of each class. These boundaries will segregate RC from GS. In the same way, let's try to see the effect of value "K" on the class boundaries. The following are the different boundaries

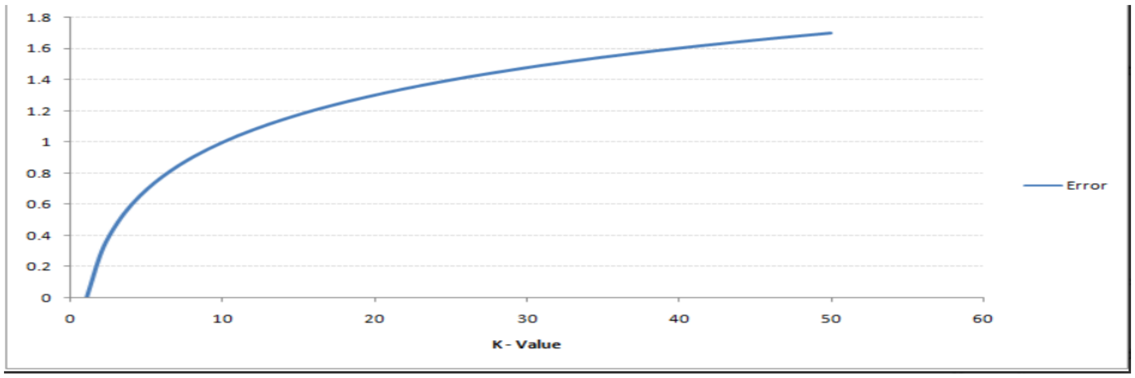
As shown in Figure 3



(separating the two classes with different values of K.) Figer3

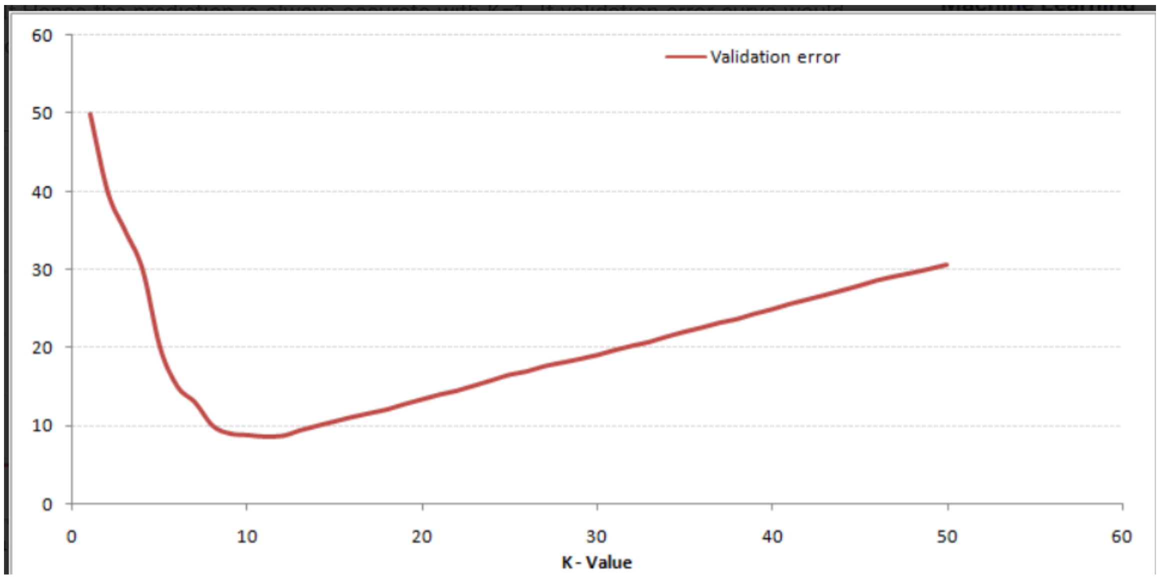


If you watch carefully, you can see that the boundary becomes smoother with increasing value of K. With K increasing to infinity it finally becomes all blue or all red depending on the total majority. The training error rate and the validation error rate are two parameters we need to access different K-value. Following is the



(curve for the training error rate with a varying value of K.)

As you can see, the error rate at K=1 is always zero for the training sample. This is because the closest point to any training data point is itself.Hence the prediction is always accurate with K=1. If validation error curve would have been similar, our choice of K would have been 1. Following is the validation error curve with varying value of K:



This makes the story more clear. At K=1, we were overfitting the boundaries. Hence, error rate initially decreases and reaches a minima. After the minima point, it then increase with increasing K. To get the optimal value of K, you can segregate the training and validation from the initial dataset. Now plot the validation error curve to get the optimal value of K. This value of K should be used for all predictions.[2]

5-The two primary disadvantages

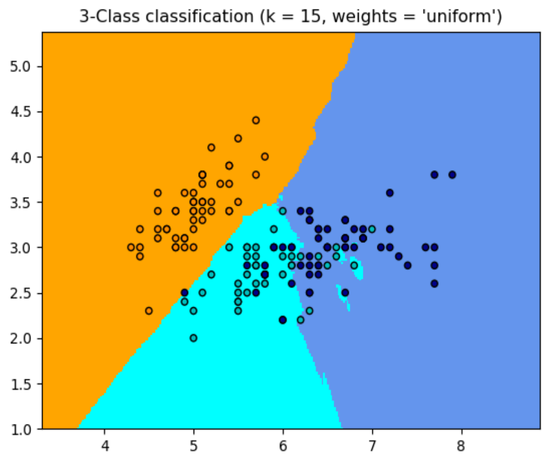
1. k-NN doesn’t work well with non--numeric predictor values

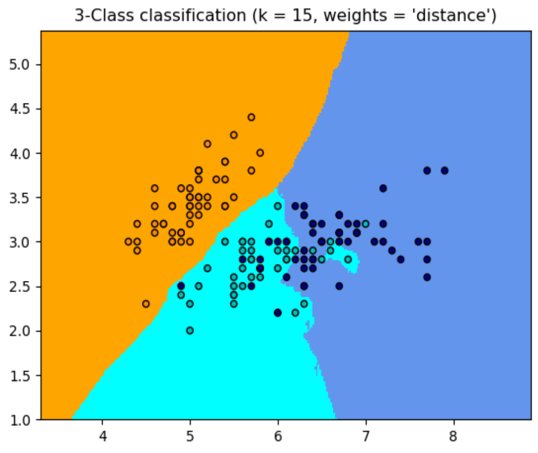
2. **Does not work well with large dataset:** In large datasets, the cost of calculating the distance between the new point and each existing points is huge which degrades the performance of the algorithm.

3. **Need feature scaling:** We need to do feature scaling (standardization and normalization) before applying KNN algorithm to any dataset. If we don't do so, KNN may generate wrong predictions.[6]

6-Nearest Neighbors Classification

Neighbors-based classification is a type of *instance-based learning* or *non-generalizing learning*: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.scikit-learn implements two different nearest neighbors classifiers: **KNeighborsClassifier** implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user. **RadiusNeighborsClassifier** implements learning based on the number of neighbors within a fixed radius r of each training point, where r is a floating-point value specified by the user.The k-neighbors classification in **KNeighborsClassifier** is the most commonly used technique. The optimal choice of the value k is highly data-dependent: in general a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.In cases where the data is not uniformly sampled, radius-based neighbors classification in **RadiusNeighborsClassifier** can be a better choice. The user specifies a fixed radius r, such that points in sparser neighborhoods use fewer nearest neighbors for the classification. For high-dimensional parameter spaces, this method becomes less effective due to the so-called “curse of dimensionality”.The basic nearest neighbors classification uses uniform weights: that is, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. Under some circumstances, it is better to weight the neighbors such that nearer neighbors contribute more to the fit. This can be accomplished through the weights keyword. The default value, weights = 'uniform', assigns uniform weights to each neighbor. weights = 'distance' assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied to compute the weights.[1]





7-KNN vs K-mean

Many people get confused between these two statistical techniques- K-mean and K-nearest neighbor. See some of the difference below.

1. **K-mean** is a clustering technique which tries to split data points into K-clusters such that the points in each cluster tend to be near each other whereas K-nearest neighbor tries to determine the classification of a point, combines the classification of the K nearest points [5]

2. **K-mean** is an unsupervised learning technique (no dependent variable) whereas KNN is a supervised learning algorithm (dependent variable exists)[5]

8-Conclusion

KNN is an effective machine learning algorithm that can be used in credit scoring, prediction of cancer cells, image recognition, and many other applications. The main importance of using KNN is that it's easy to implement and works well with small datasets. In this study, by using classifiers of the data KNN…

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).[4]

References

- https://scikit-learn.org/stable/modules/neighbors.html#unsupervised-nearest-neighbors
- https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/
- https://blog.quantinsti.com/machine-learning-k-nearest-neighbors-knn-algorithm-python/#:~:text=K%2DNearest%20Neighbors%20(KNN)%20is%20one%20of%20the%20simplest,majority%20vote%20to%20its%20neighbors.
- https://medium.com/@chiragsehra42/k-nearest-neighbors-explained-easily-c26706aa5c7f
- https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html
- http://theprofessionalspoint.blogspot.com/2019/02/advantages-and-disadvantages-of-knn.html

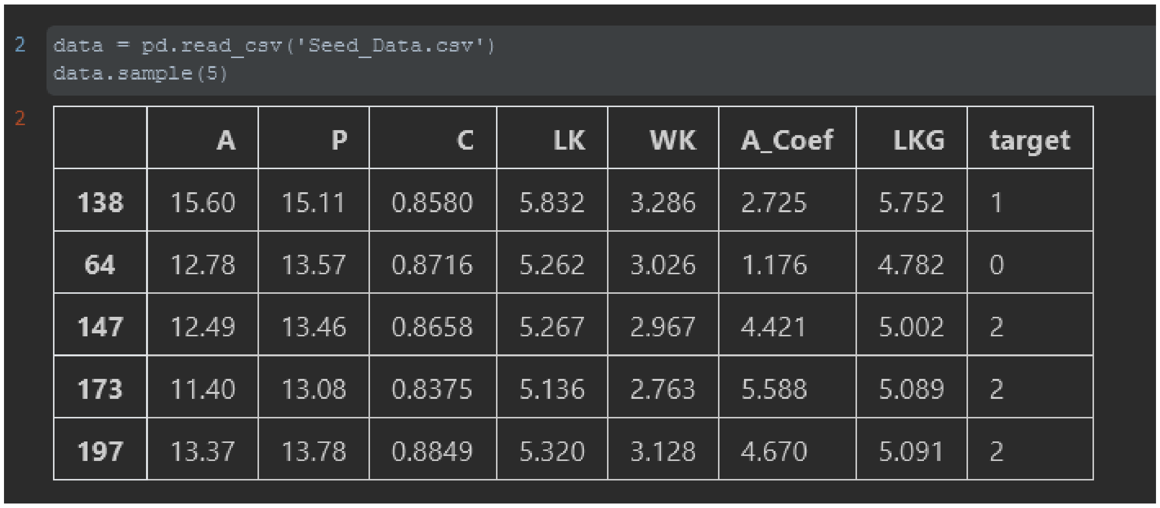
KNN Algorithm

1- Import The libraries

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns; sns.set(style="ticks", color_codes=True)
```

2- Read data from file

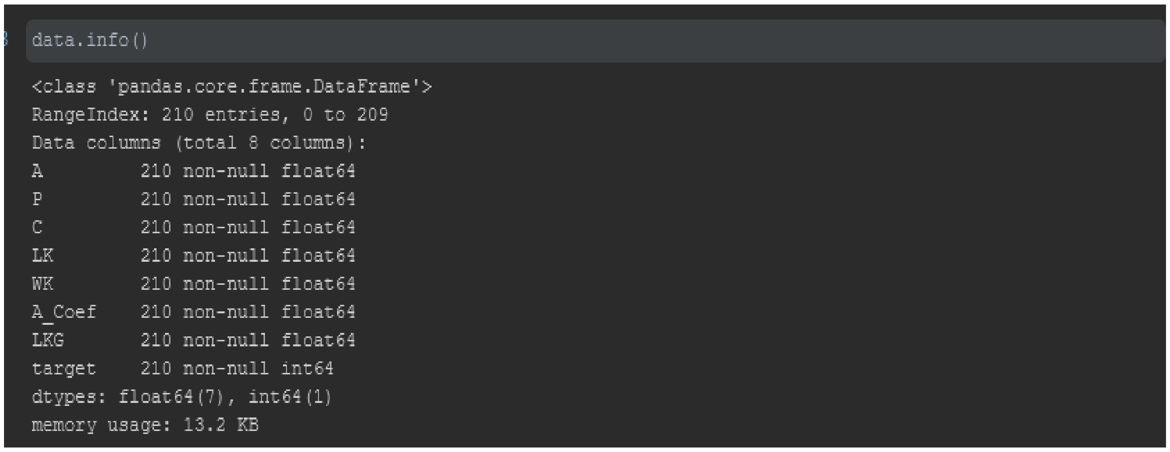
```
data = pd.read_csv('Seed_Data.csv')
data.sample(5)
```



```
data = pd.read_csv('Seed_Data.csv')
data.sample(5)
```

	A	P	C	LK	WK	A_Coef	LKG	target
138	15.60	15.11	0.8580	5.832	3.286	2.725	5.752	1
64	12.78	13.57	0.8716	5.262	3.026	1.176	4.782	0
147	12.49	13.46	0.8658	5.267	2.967	4.421	5.002	2
173	11.40	13.08	0.8375	5.136	2.763	5.588	5.089	2
197	13.37	13.78	0.8849	5.320	3.128	4.670	5.091	2

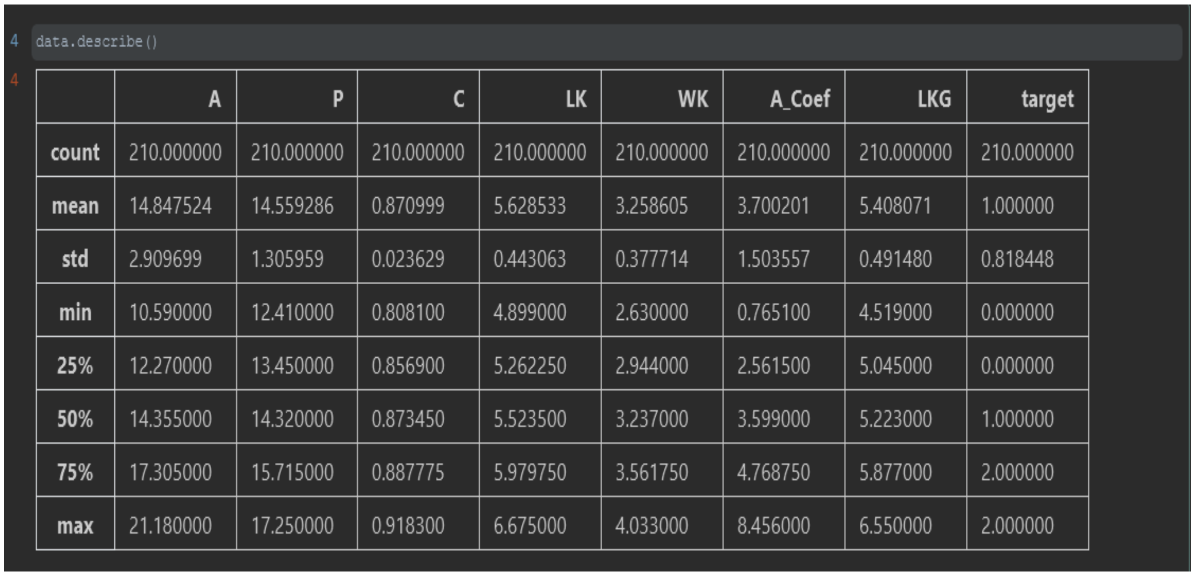
3- data.info()



```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210 entries, 0 to 209
Data columns (total 8 columns):
A          210 non-null float64
P          210 non-null float64
C          210 non-null float64
LK         210 non-null float64
WK         210 non-null float64
A_Coef     210 non-null float64
LKG        210 non-null float64
target     210 non-null int64
dtypes: float64(7), int64(1)
memory usage: 13.2 KB
```

4-data.describe()

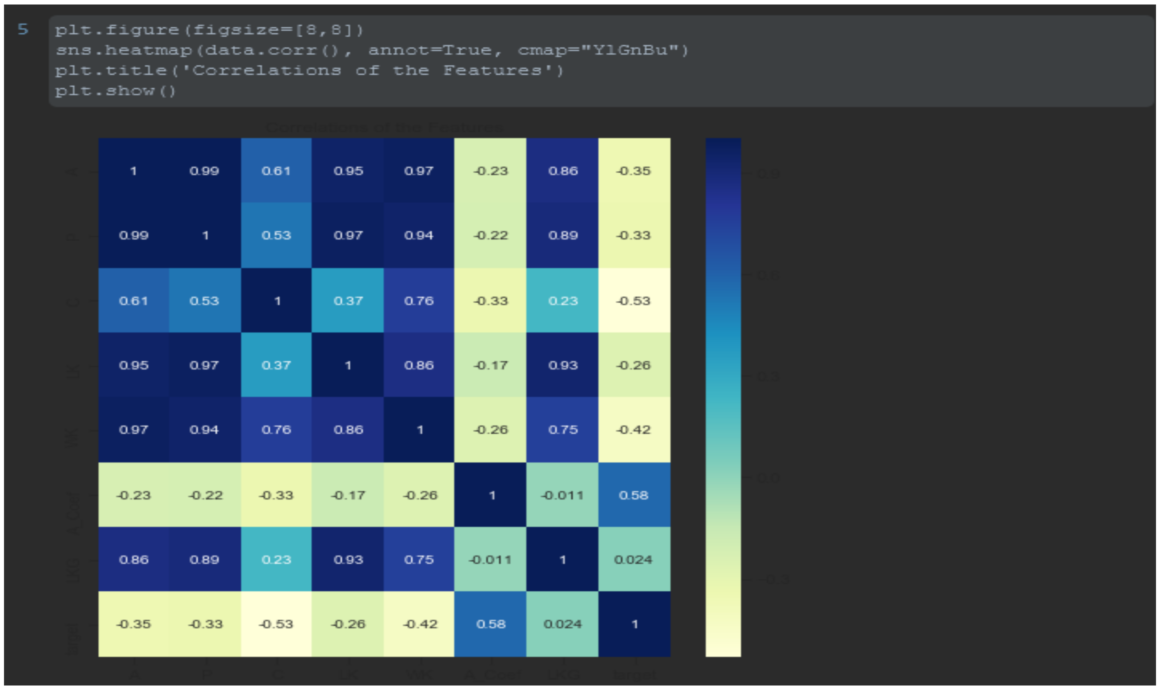


```
data.describe()
```

	A	P	C	LK	WK	A_Coef	LKG	target
count	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000
mean	14.847524	14.559286	0.870999	5.628533	3.258605	3.700201	5.408071	1.000000
std	2.909699	1.305959	0.023629	0.443063	0.377714	1.503557	0.491480	0.818448
min	10.590000	12.410000	0.808100	4.899000	2.630000	0.765100	4.519000	0.000000
25%	12.270000	13.450000	0.856900	5.262250	2.944000	2.561500	5.045000	0.000000
50%	14.355000	14.320000	0.873450	5.523500	3.237000	3.599000	5.223000	1.000000
75%	17.305000	15.715000	0.887775	5.979750	3.561750	4.768750	5.877000	2.000000
max	21.180000	17.250000	0.918300	6.675000	4.033000	8.456000	6.550000	2.000000

5-Visualization

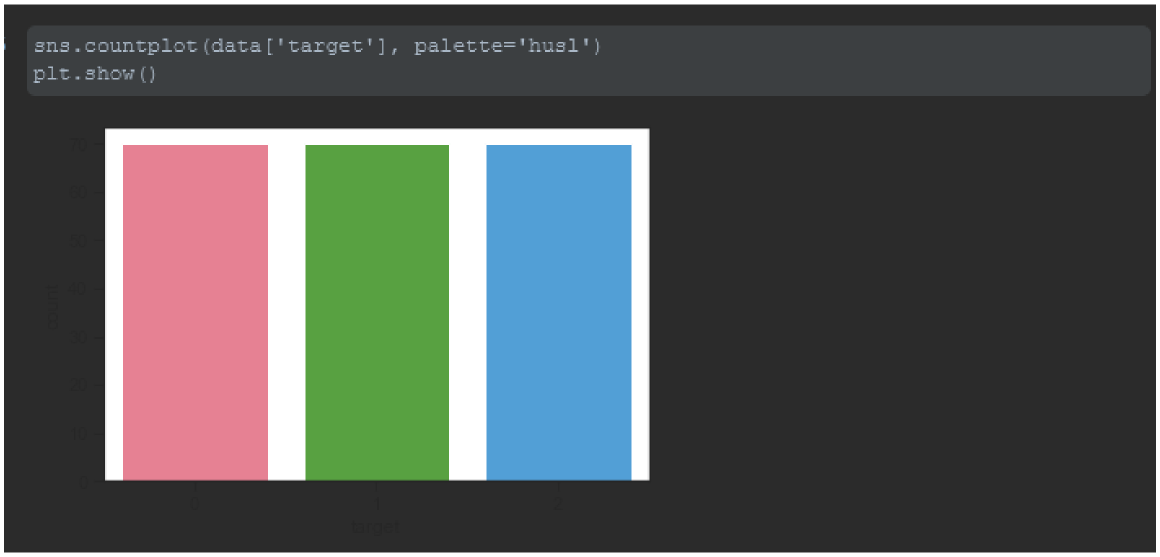
plt.figure(figsize=[8,8])
sns.heatmap(data.corr(), annot=True, cmap="YlGnBu")
plt.title('Correlations of the Features')
plt.show()



```
plt.figure(figsize=[8,8])
sns.heatmap(data.corr(), annot=True, cmap="YlGnBu")
plt.title('Correlations of the Features')
plt.show()
```

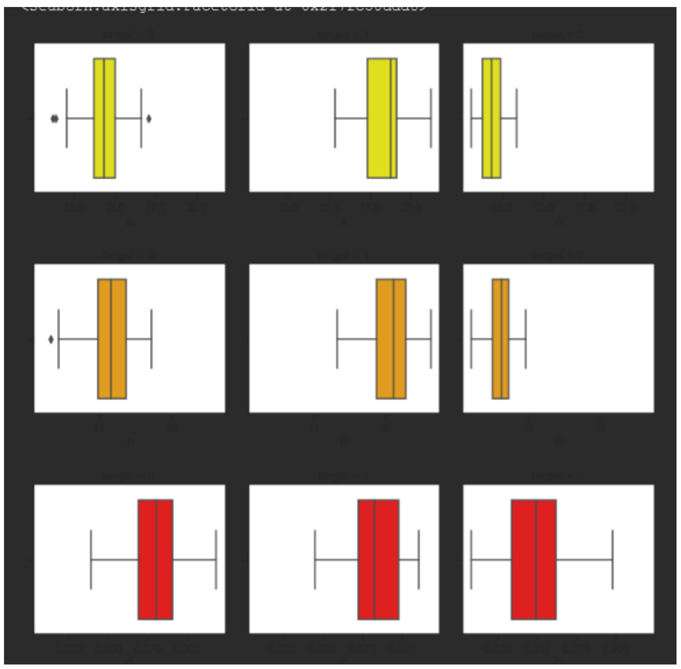
6- Values

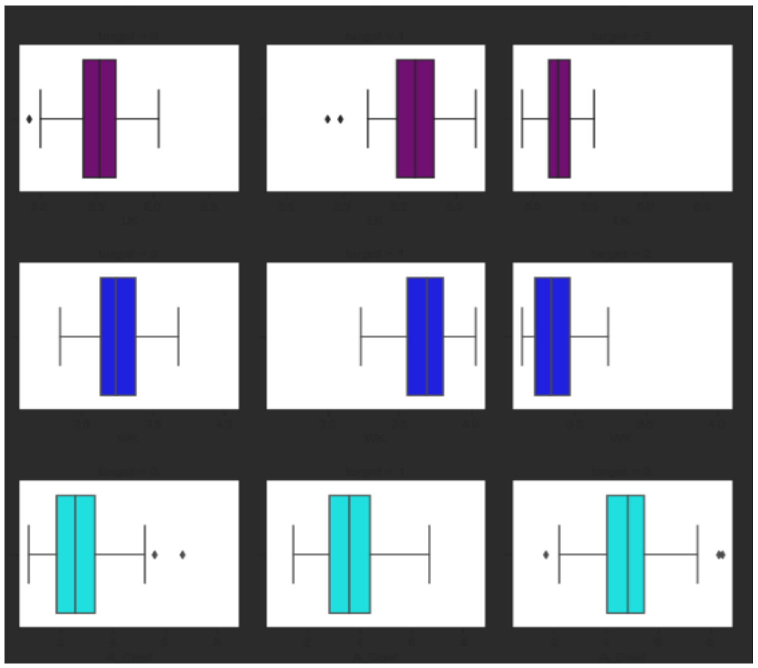
```
sns.countplot(data['target'], palette='husl')
plt.show()
```



```
sns.countplot(data['target'], palette='husl')
plt.show()
```

```
a = sns.FacetGrid(data, col='target')
a.map(sns.boxplot, 'A', color='yellow', order=['0', '1', '2'])

p = sns.FacetGrid(data, col='target')
p.map(sns.boxplot, 'P', color='orange', order=['0', '1', '2'])

c = sns.FacetGrid(data, col='target')
c.map(sns.boxplot, 'C', color='red', order=['0', '1', '2'])

lk = sns.FacetGrid(data, col='target')
lk.map(sns.boxplot, 'LK', color='purple', order=['0', '1', '2'])

wk = sns.FacetGrid(data, col='target')
wk.map(sns.boxplot, 'WK', color='blue', order=['0', '1', '2'])

acoef = sns.FacetGrid(data, col='target')
acoef.map(sns.boxplot, 'A_Coef', color='cyan', order=['0', '1', '2'])

lkg = sns.FacetGrid(data, col='target')
lkg.map(sns.boxplot, 'LKG', color='green', order=['0', '1', '2'])
```





7- Split-out validation dataset 80 train , 20 test

```
array = data.values
x = array[:,0:7]
y = array[:,7]
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)
```

8- KNN Applied

```
from sklearn.neighbors import KNeighborsClassifier
import math
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(x_train, y_train)
pred = knn.predict(x_test)
print(round(float(sum(pred==y_test)/len(y_test)),2))
```

```
0.88
```