Advance Machine Learning

Project

He NI

BI2

efrei PARIS
Ecole d'ingénieurs du numérique

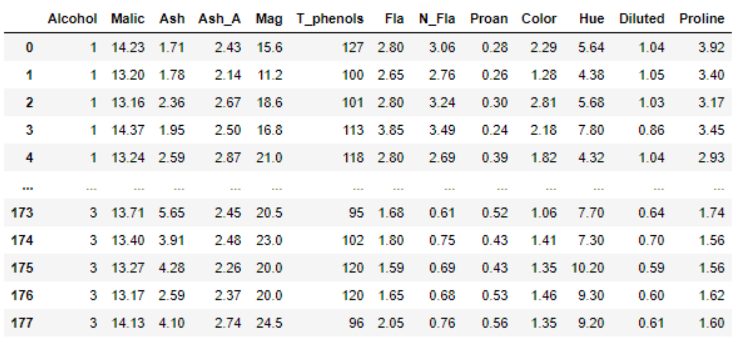
Agenda

efrei PARIS
Ecole d'ingénieurs du numérique

1. Analysis of the dataset:

The dataset we choose is wine

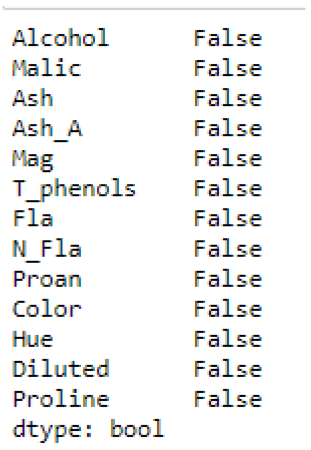
We import the downloaded data into python



	Alcohol	Malic	Ash	Ash_A	Mag	T_phenols	Fla	N_Fla	Proan	Color	Hue	Diluted	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93
...	...	...	...	...	...	...	...	...	...	...	...	...	...
173	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64	1.74
174	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56
175	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56
176	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62
177	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60

Check for missing values in the data

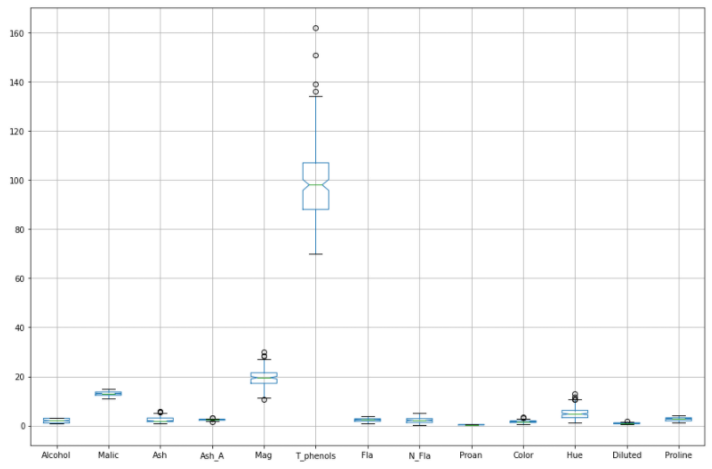
Check for null values in the dataset



```
Alcohol      False
Malic        False
Ash          False
Ash_A        False
Mag          False
T_phenols    False
Fla          False
N_Fla        False
Proan        False
Color        False
Hue          False
Diluted      False
Proline      False
dtype: bool
```

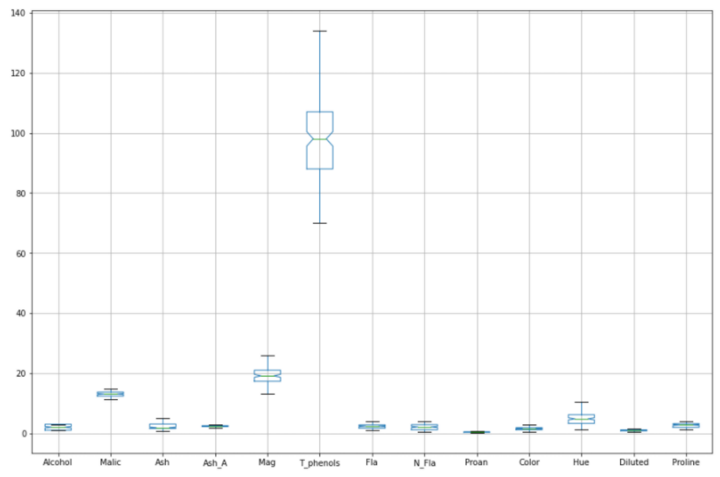
Check outliers

Checking the outliers in the dataset, we use a box plot and can see that the dots in the plot are the outliers values. All we have to do is remove them.

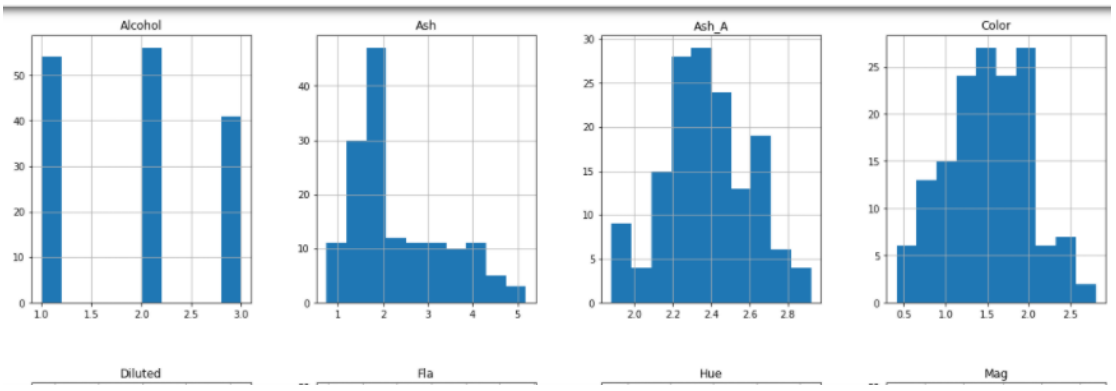


#Q1=25%
#Q3=75%
#IQR = Q3-Q1
#UpLimit：Q3+1.5IQR
#DownLimit：Q1-1.5IQR
Here I cycled through the deletion process 3 times, as some outliers would not be clean if deleted only once.

```
<matplotlib.axes._subplots.AxesSubplot at 0x2713717bdf0>
```

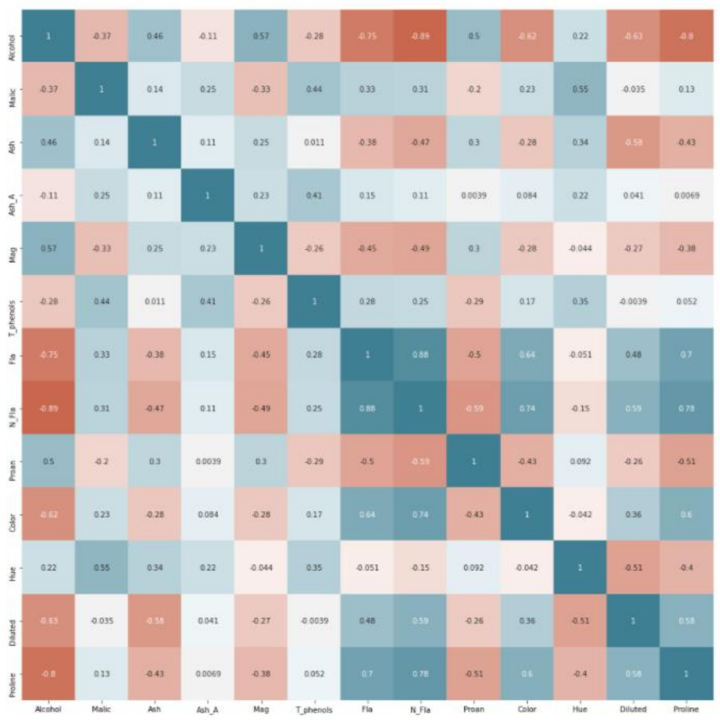


Feature Visualization



Correlation between features

Here I have used a hotspot diagram to show the correlation between individual features.



2. Prediction:

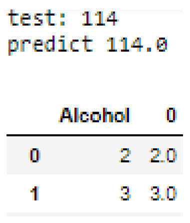
We use the first column feature "Alcohol" as the label class

Preprocessing

We first scaler the 13 features, followed by partitioning the data into a training set and a test set.

Linear regression

Here we use a linear regression algorithm.
There is a big problem here.
I want to find the dataitem in the first and second columns that are equal, count them, and then use this count/sum to get an accuracy rate
But I found that the predicted value of pred_y using linear regression, which is of type float64 and should be rounded, ex: the first row is 2.0 and the true value is 1.98888888.
I found this situation also when I tried to convert the type of pred_y to an int type using astype and got a completely different value than before the conversion.
In the end I couldn't come up with a good solution, so I ended up using the (sum) method.
The accuracy rate is 1

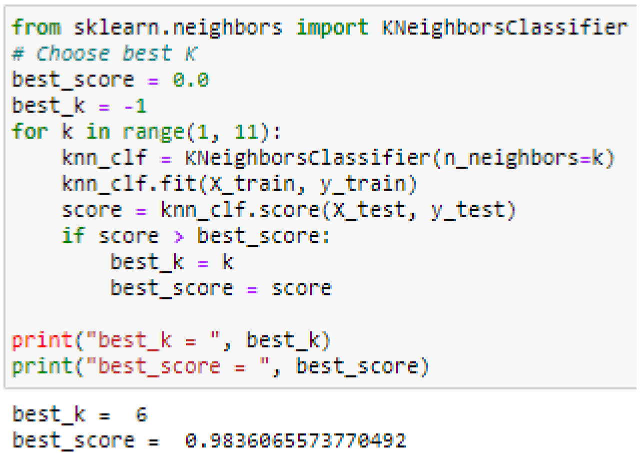


```
test: 114
predict 114.0
```

	Alcohol	0
0	2	2.0
1	3	3.0

KNN

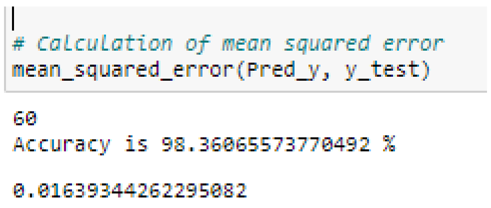
We use the knn algorithm with hyperparameter tuning prior to prediction.



```
from sklearn.neighbors import KNeighborsClassifier
# Choose best K
best_score = 0.0
best_k = -1
for k in range(1, 11):
    knn_clf = KNeighborsClassifier(n_neighbors=k)
    knn_clf.fit(X_train, y_train)
    score = knn_clf.score(X_test, y_test)
    if score > best_score:
        best_k = k
        best_score = score

print("best_k = ", best_k)
print("best_score = ", best_score)
```

```
best_k =  6
best_score =  0.9836065573770492
```

We conclude that we can get the highest accuracy when K is taken as 6.
We calculated his mean squared error.
As you can see, the error is very small, which means our model is good.



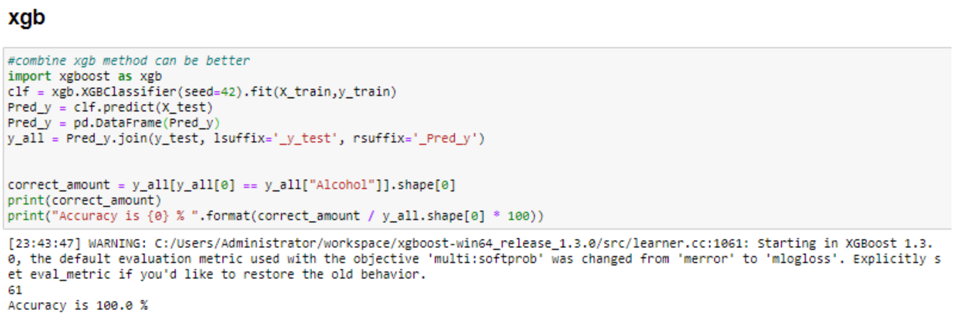
```
# Calculation of mean squared error
mean_squared_error(Pred_y, y_test)
```

```
60
Accuracy is 98.36065573770492 %

0.01639344262295082
```

XGB

We should provide a solution to improve the results, for example by combining multiple methods. So I came up with XGB

The Xgboost correspondence is a bunch of CART trees. The idea of the algorithm is to keep adding trees and keep feature splitting to grow a tree. Each time a tree is added, it is actually learning a new function to fit the residuals of the last prediction. When we finish training and get k trees, we want to predict the score of a sample, which is actually based on the characteristics of the sample, in each tree will fall to a corresponding leaf node, each leaf node corresponds to a score, and finally just add up the corresponding scores of each tree is the predicted value of the sample.



xgb

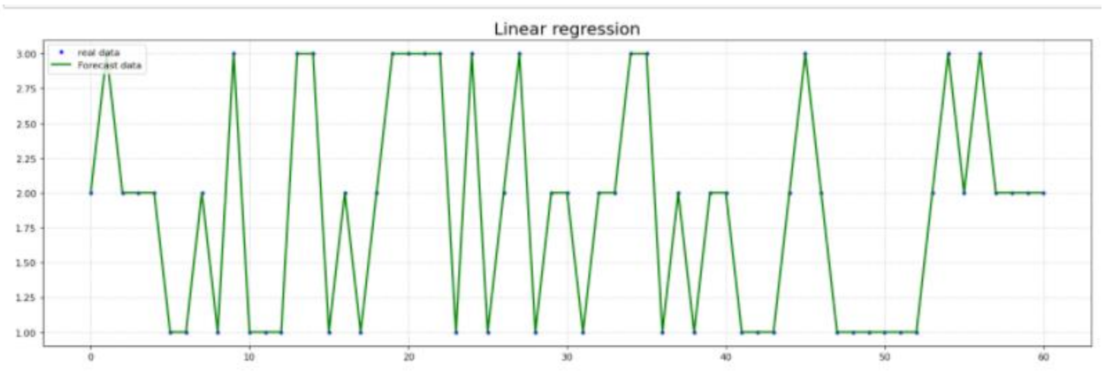
```
#combine xgb method can be better
import xgboost as xgb
clf = xgb.XGBClassifier(seed=42).fit(X_train,y_train)
Pred_y = clf.predict(X_test)
Pred_y = pd.DataFrame(Pred_y)
y_all = Pred_y.join(y_test, lsuffix='_y_test', rsuffix='_Pred_y')


correct_amount = y_all[y_all[0] == y_all["Alcohol"]].shape[0]
print(correct_amount)
print("Accuracy is {0} % ".format(correct_amount / y_all.shape[0] * 100))
```

```
[23:43:47] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
61
Accuracy is 100.0 %
```

The accuracy of XGB is 100%

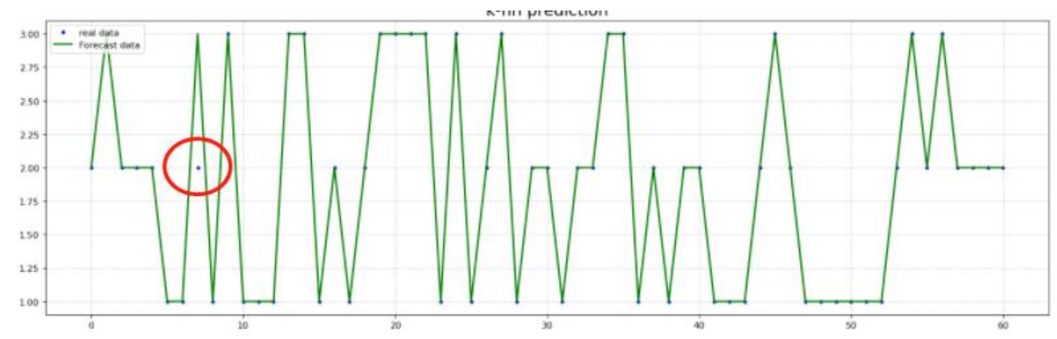
3. Visualization

Linear regression visualization



The green line is the predicted data, the blue dots are the raw data, as you can see the accuracy rate is 100%

KNN visualization



Knn's accuracy rate is 98%, with one point not met

4.Theoretical details

For this project, we used both linear regression and K-NN prediction methods.

Linear regression is a statistical analysis method that uses regression analysis in mathematical statistics to determine the quantitative relationship between two or more variables that depend on each other, and is widely used. The expression takes the form

$$y = w'x + e$$

with e being the error following a normal distribution with mean 0.

The so-called K nearest neighbours, meaning K nearest neighbours, says that each sample can be represented by its closest K neighbouring values. The nearest neighbour algorithm is a way of classifying each record in a data set
The distance between points we use the Euclidean distance.

$$d_2(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

The value of k needs to be determined by ourselves and is generally taken to be an odd number. Earlier I also used the hyperparameter adjustment method to work out that k is best taken as 6.