資料分析與學習基石

Homework 1 – First visit in Kaggle data

醫工 113 虎冠廷

(一)基本資料集描述：

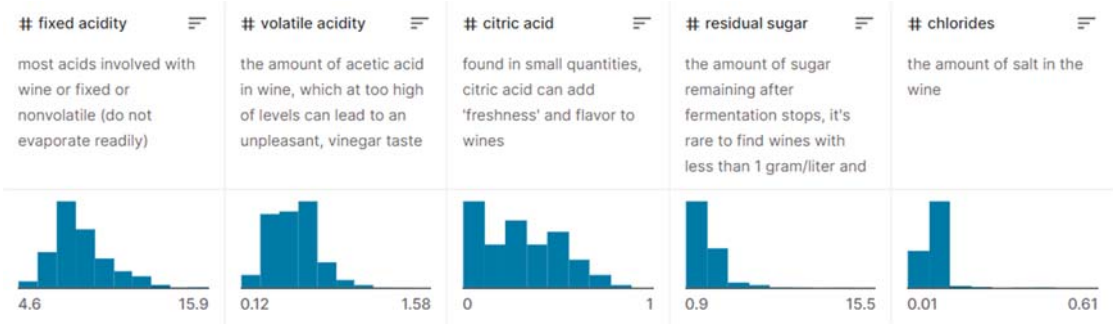
資料集名稱：Red Wine Quality

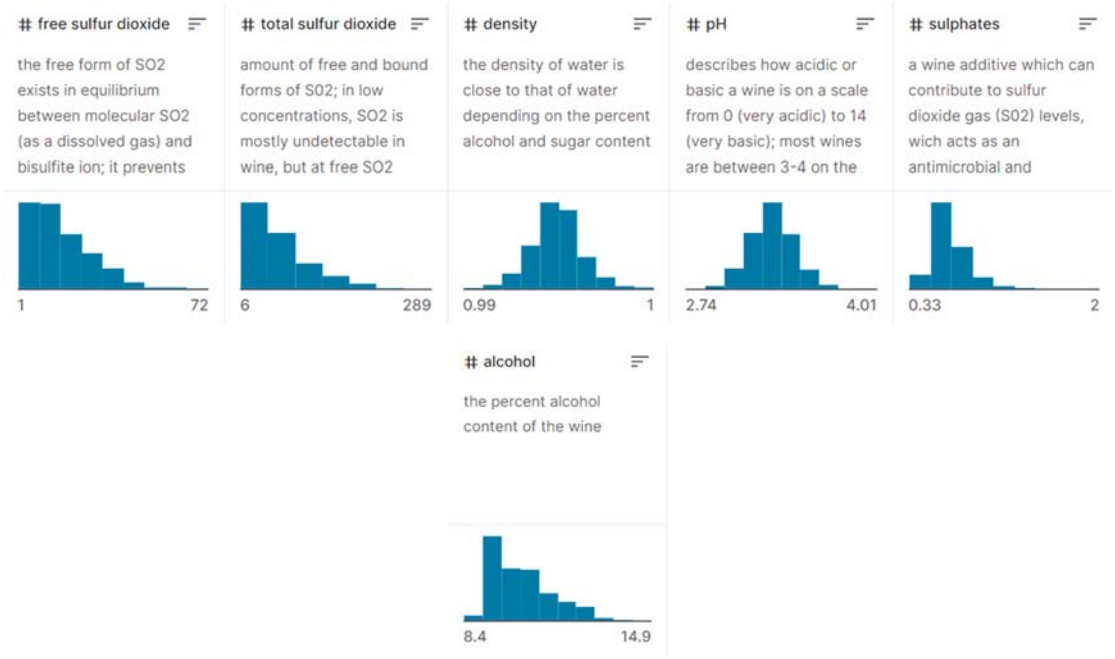
1. 資料集介紹：這個數據集與葡萄牙“ Vinho Verde ”葡萄酒的紅酒和白酒有關，目標是根據物理化學測試對葡萄酒質量進行建模。詳細信息可以參閱參考文獻 [1]。由於隱私和物流問題，只有物理化學變量（輸入）和感官（輸出）變量可用（例如，沒有關於葡萄類型、葡萄酒品牌、葡萄酒售價等的數據）。這些數據集可以被視為分類或回歸任務。等級是有序的，不平衡（例如，普通葡萄酒比優秀或差的葡萄酒多得多）。

2. 資料欄位：

Input variables (based on physicochemical tests):

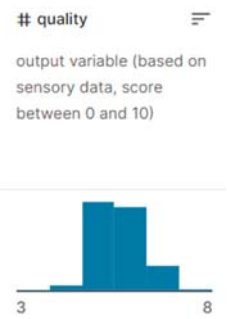
1 - fixed acidity：固定酸度
2 - volatile acidity：揮發性酸度
3 - citric acid：檸檬酸
4 - residual sugar：殘糖
5 - chlorides：氯化物
6 - free sulfur dioxide：游離二氧化硫
7 - total sulfur dioxide：總二氧化硫
8 - density：密度
9 - pH：酸鹼值
10 - sulphates：硫酸鹽
11 - alcohol：酒精





Output variable (based on sensory data):

12 - quality ：質量（分數在 0 到 10 之間）



(二)資料特性與分析：

根據作者整理出來的資料欄位，資料分析後最主要的輸出是葡萄酒的品質，除了使用回歸模型之外，還有一些其他標準可以分析，以商業角度來審視，最有可能且最有價值的例子是設立一個商品好壞的截止值，例如 7 或更高分被歸類為“1”（好），其餘為“0”（不好）。

使用機器學習來確定哪些物理化學特性使葡萄酒“好”，這就是這份資料集的特性及分析後能得到的價值。

（三）Code 方法與比較 1:

1.套件 Package:

pandas → 用於表格數據操作

seaborn、matplotlib→資料視覺化

sklearn→用於機器學習演算法

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
%matplotlib inline
```

2. 載入資料集：

此步驟以 pandas 讀取 csv 格式的 dataset

```
#Loading dataset
wine = pd.read_csv('../input/winequality-red.csv')
```

3. 查看資料集分布及資料型態：

以 pandas 簡單查看資料集

```
#Let's check how the data is distributed
wine.head()
```

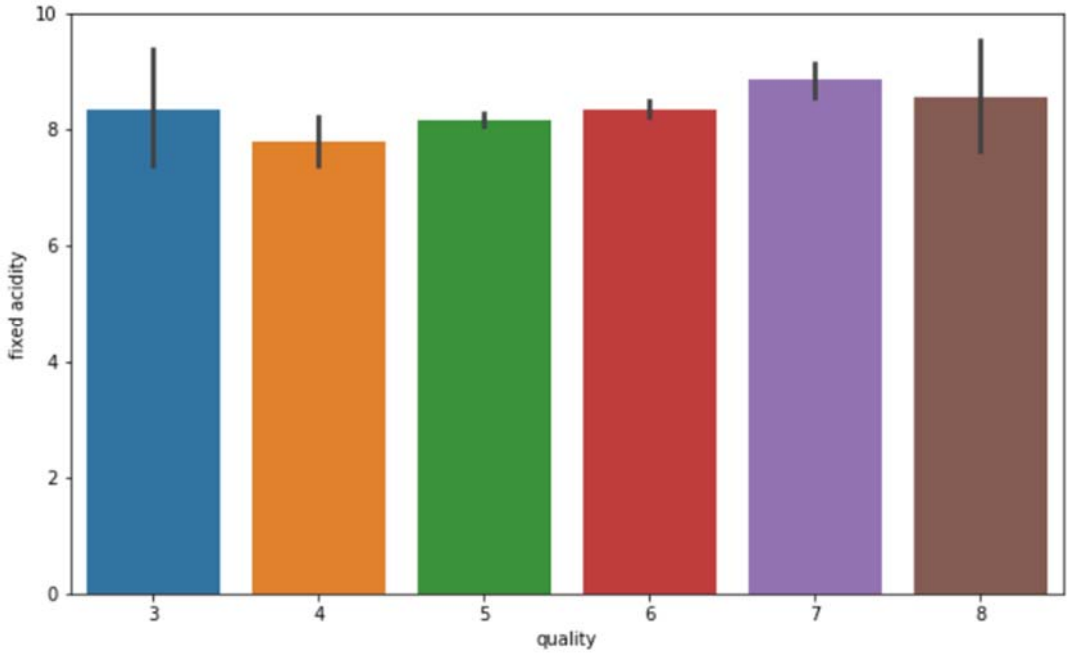
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

```
#Information about the data columns
wine.info()
```

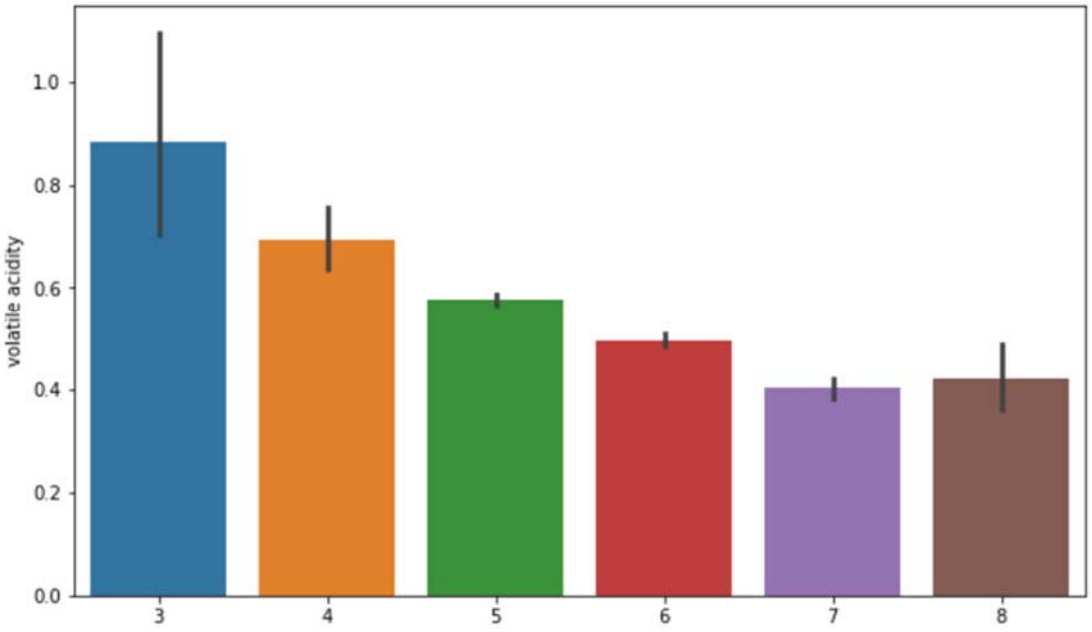
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
fixed acidity           1599 non-null float64
volatile acidity        1599 non-null float64
citric acid             1599 non-null float64
residual sugar          1599 non-null float64
chlorides               1599 non-null float64
free sulfur dioxide     1599 non-null float64
total sulfur dioxide    1599 non-null float64
density                 1599 non-null float64
pH                      1599 non-null float64
sulphates               1599 non-null float64
alcohol                 1599 non-null float64
quality                 1599 non-null int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

4. 使用圖表觀察各數據之間之關聯：

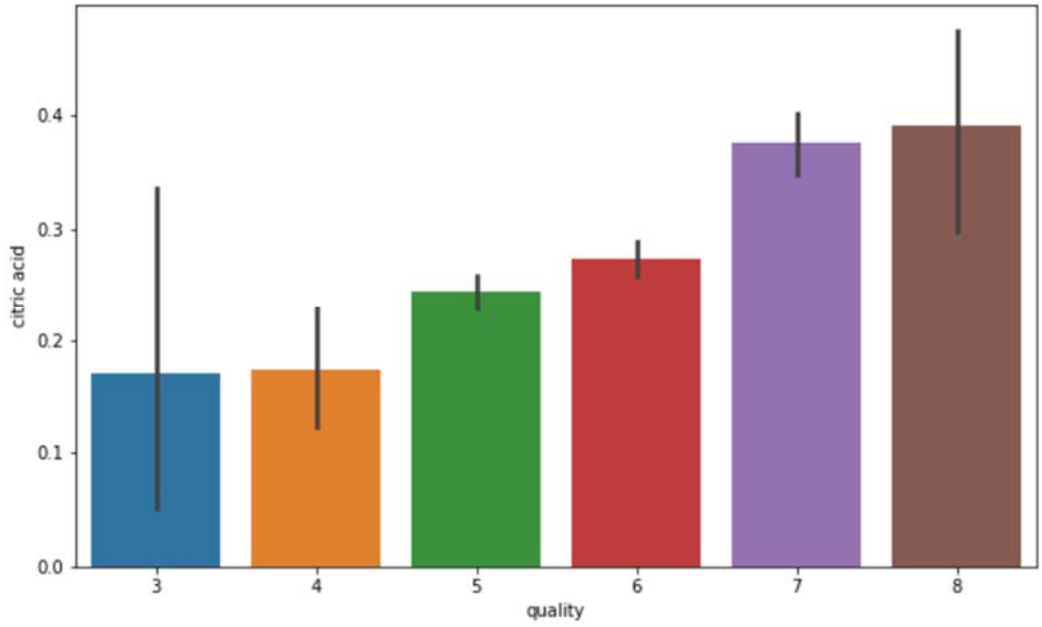
```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'fixed acidity', data = wine)
```



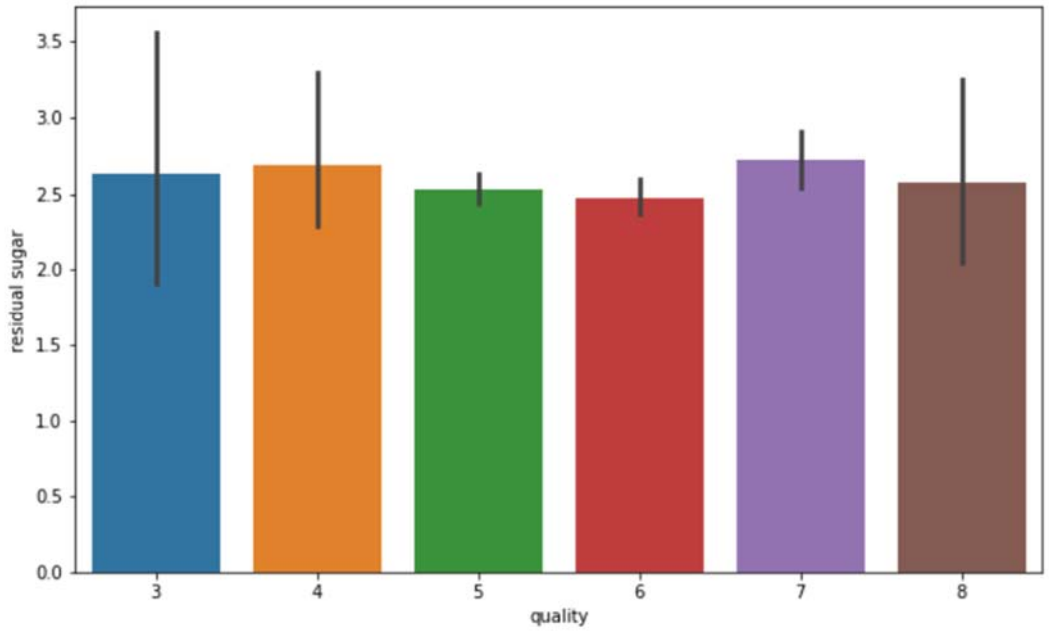

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'volatile acidity', data = wine)
```



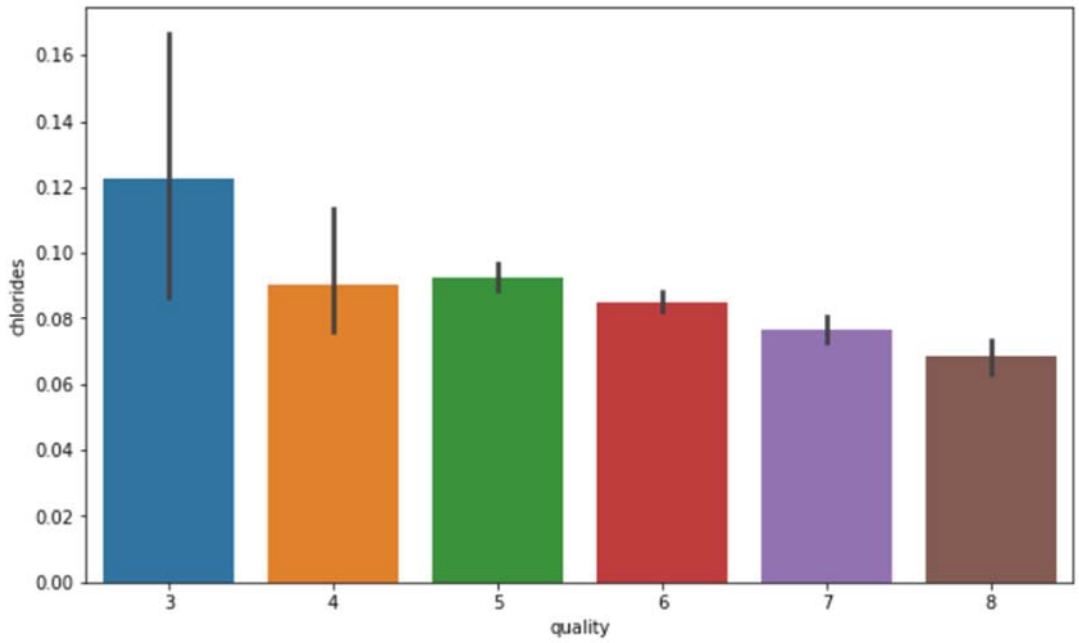

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'citric acid', data = wine)
```



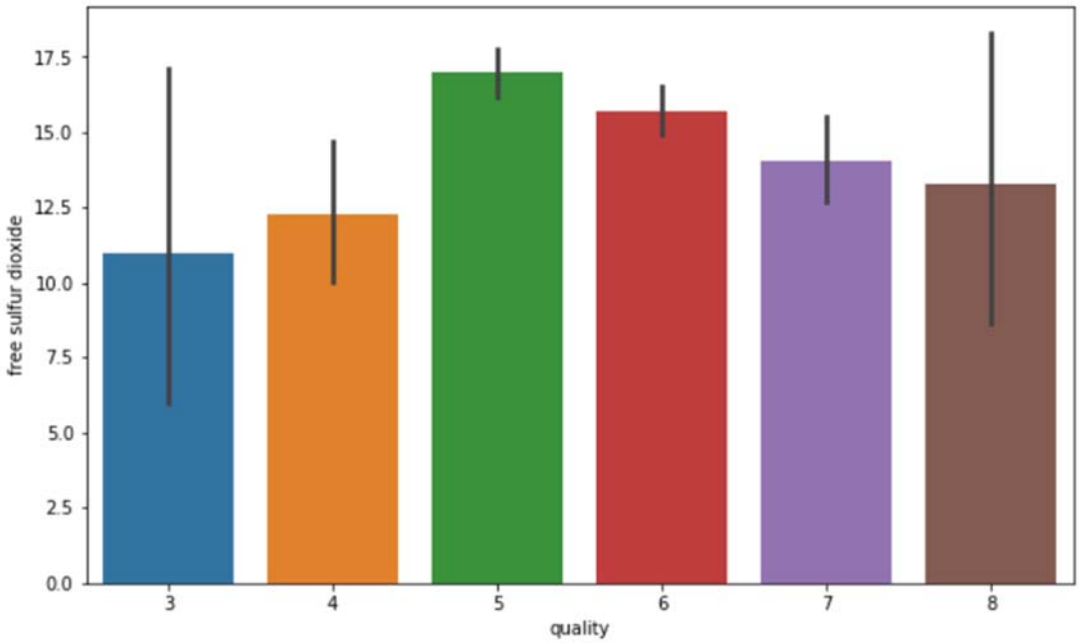

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'residual sugar', data = wine)
```



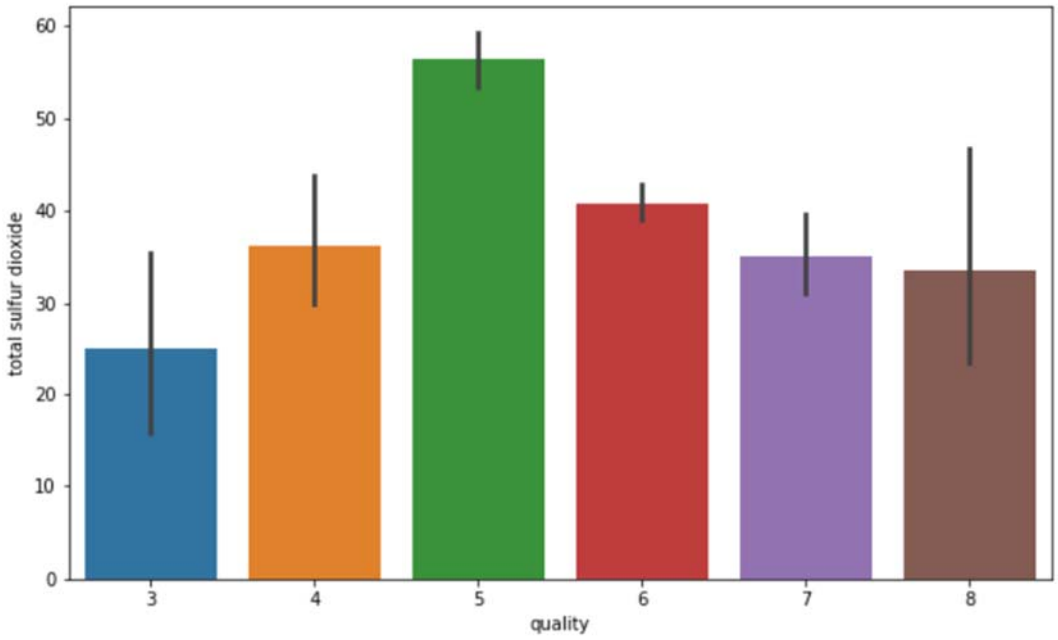

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'chlorides', data = wine)
```



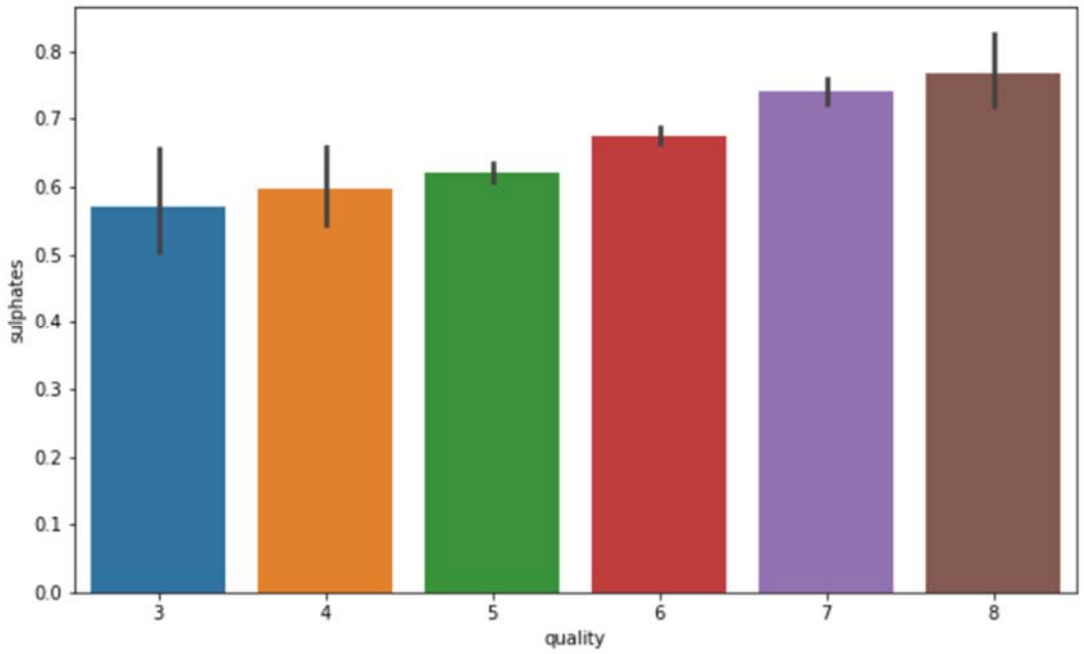

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'free sulfur dioxide', data = wine)
```



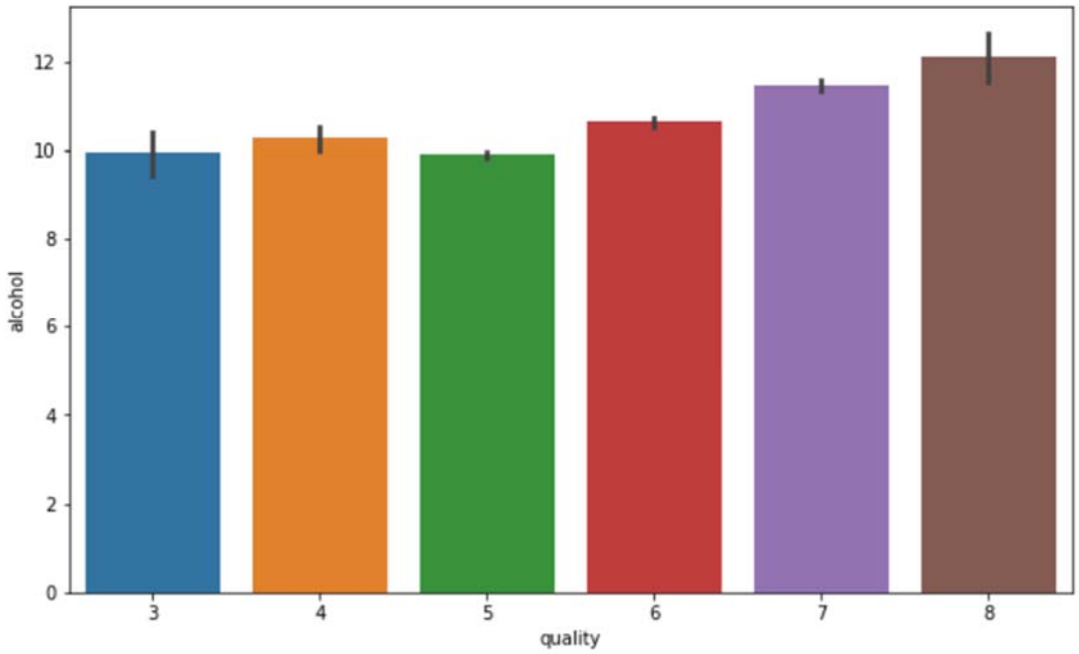

```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'total sulfur dioxide', data = wine)
```




```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'sulphates', data = wine)
```




```
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'alcohol', data = wine)
```



從圖表中可以輕鬆得知各(輸入)數據和結果的關聯。

正相關：citric acid、sulphates、alcohol

負相關：volatile acidity、chlorides

5. 資料預處理：

透過資料預處理，使數據能進行機器學習對反應變量進行二項分類、通過給出質量限制來區分葡萄酒的好壞。

```
bins = (2, 6.5, 8)
group_names = ['bad', 'good']
wine['quality'] = pd.cut(wine['quality'], bins = bins, labels = group_names)
```

```
label_quality = LabelEncoder()
```

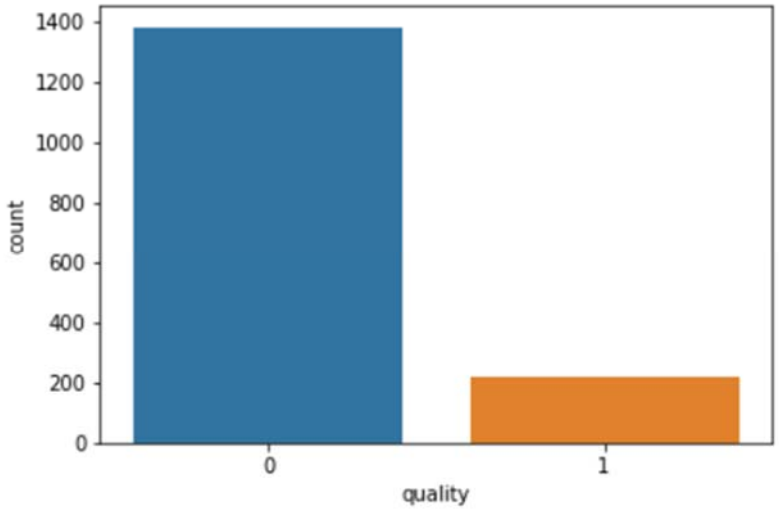
壞的變成 0，好的變成 1。

```
wine['quality'] = label_quality.fit_transform(wine['quality'])
```

```
wine['quality'].value_counts()
```

```
0    1382
1     217
Name: quality, dtype: int64
```

```
sns.countplot(wine['quality'])
```



分割數據集為反應變量和特徵變量

```
X = wine.drop('quality', axis = 1)
y = wine['quality']
```

分割數據集為 Training 和 Testing

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

應用標準縮放以獲得優化結果

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

將 Training 資料集和 Testing 資料集導入 Machine Learning

- precision：模型預測為陽性樣本預測正確的比例
- recall：事實為陽性樣本且被預測為陽性之比例
- F1 值：精確率和召回率之調和平均數，可以衡量模型準確度

6. 套用模型進行訓練：

- Random Forest Classifier

```
rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
pred_rfc = rfc.predict(X_test)
```

顯示模型表現：

```
print(classification_report(y_test, pred_rfc))
```

	precision	recall	f1-score	support
0	0.90	0.96	0.93	273
1	0.66	0.40	0.50	47
avg / total	0.87	0.88	0.87	320

Random Forest Classifier → Accuracy = 87%

顯示 confusion matrix

```
print(confusion_matrix(y_test, pred_rfc))
```

```
[[263  10]
 [ 28  19]]
```

- Stochastic Gradient Decent Classifier

```
sgd = SGDClassifier(penalty=None)
sgd.fit(X_train, y_train)
pred_sgd = sgd.predict(X_test)
```

顯示模型表現：

```
print(classification_report(y_test, pred_sgd))
```

	precision	recall	f1-score	support
0	0.91	0.89	0.90	273
1	0.44	0.51	0.47	47
avg / total	0.84	0.83	0.84	320

Stochastic Gradient Decent Classifier → Accuracy = 84%

顯示 Confusion Matrix

```
print(confusion_matrix(y_test, pred_sgd))
```

```
[[242  31]
 [ 23  24]]
```

- Support Vector Classifier

```
svc = SVC()
svc.fit(X_train, y_train)
pred_svc = svc.predict(X_test)
```

顯示模型表現：

```
print(classification_report(y_test, pred_svc))
```

	precision	recall	f1-score	support
0	0.88	0.98	0.93	273
1	0.71	0.26	0.37	47
avg / total	0.86	0.88	0.85	320

Support Vector Classifier → Accuracy = 86%

7. 提高模型的準確性：

- Grid Search CV

Grid Search CV 函數會自動作 Cross Validation，並且統計準確率的平均數/標準差，幫我們找出最佳參數組合。

```
param = {
    'C': [0.1,0.8,0.9,1,1.1,1.2,1.3,1.4],
    'kernel':['linear', 'rbf'],
    'gamma' :[0.1,0.8,0.9,1,1.1,1.2,1.3,1.4]
}
grid_svc = GridSearchCV(svc, param_grid=param, scoring='accuracy', cv=10)
```

```
grid_svc.fit(X_train, y_train)
```

```
GridSearchCV(cv=10, error_score='raise',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'C': [0.1, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4], 'kernel': ['linear', 'rbf'],
'gamma': [0.1, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='accuracy', verbose=0)
```

顯示 SVC model 的最佳參數

```
grid_svc.best_params_
```

```
{'C': 1.2, 'gamma': 0.9, 'kernel': 'rbf'}
```

使用最佳參數重跑 SVC model 並顯示結果

```
svc2 = SVC(C = 1.2, gamma =  0.9, kernel= 'rbf')
svc2.fit(X_train, y_train)
pred_svc2 = svc2.predict(X_test)
print(classification_report(y_test, pred_svc2))
```

	precision	recall	f1-score	support
0	0.90	0.99	0.94	273
1	0.89	0.34	0.49	47
avg / total	0.90	0.90	0.88	320

使用 Grid Search CV，SVC 的 Accuracy 從 86% 提高到 90%

SGD 和 Random forest 的交叉驗證分數

```
rfc_eval = cross_val_score(estimator = rfc, X = X_train, y = y_train, cv = 10)
rfc_eval.mean()
```

```
0.91166338582677164
```

準確率從 87% 提高到 91%

（四）Code 方法與比較 2

1.套件 Package:

pandas → 用於表格數據操作

seaborn、matplotlib→資料視覺化

sklearn→用於機器學習演算法

imblearn→處理不平衡資料集

os→調用操作系統命令

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
import imblearn
from imblearn.over_sampling import SMOTE
import os
import warnings
warnings.filterwarnings("ignore")
print(os.listdir("../input"))
```

2. 載入資料集

```
df = pd.read_csv("../input/red-wine-quality-cortez-et-al-2009/winequality-red.csv")
df.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

3.查看資料集分布及資料型態

```python
def grab_col_names(dataframe, cat_th=10, car_th=20):
    # cat_cols, cat_but_car
    cat_cols = [col for col in dataframe.columns if dataframe[col].dtypes == "O"]
    num_but_cat = [col for col in dataframe.columns if dataframe[col].nunique() < cat_th and
                   dataframe[col].dtypes != "O"]
    cat_but_car = [col for col in dataframe.columns if dataframe[col].nunique() > car_th and
                   dataframe[col].dtypes == "O"]
    cat_cols = cat_cols + num_but_cat
    cat_cols = [col for col in cat_cols if col not in cat_but_car]

    # num_cols
    num_cols = [col for col in dataframe.columns if dataframe[col].dtypes != "O"]
    num_cols = [col for col in num_cols if col not in num_but_cat]

    print(f"Observations: {dataframe.shape[0]}")
    print(f"Variables: {dataframe.shape[1]}")
    print(f'cat_cols: {len(cat_cols)}')
    print(f'num_cols: {len(num_cols)}')
    print(f'cat_but_car: {len(cat_but_car)}')
    print(f'num_but_cat: {len(num_but_cat)}')

    return cat_cols, num_cols, cat_but_car
cat_cols, num_cols, cat_but_car = grab_col_names(df)
```

```
Observations: 1599
Variables: 12
cat_cols: 1
num_cols: 11
cat_but_car: 0
num_but_cat: 1
```

```python
df.info
```

```
<bound method DataFrame.info of      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0              7.4             0.700         0.00             1.9      0.076
1              7.8             0.880         0.00             2.6      0.098
2              7.8             0.760         0.04             2.3      0.092
3             11.2             0.280         0.56             1.9      0.075
4              7.4             0.700         0.00             1.9      0.076
...            ...               ...          ...             ...        ...
1594           6.2             0.600         0.08             2.0      0.090
1595           5.9             0.550         0.10             2.2      0.062
1596           6.3             0.510         0.13             2.3      0.076
1597           5.9             0.645         0.12             2.0      0.075
1598           6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
...                   ...                   ...      ...   ...        ...
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
1598                 18.0                  42.0  0.99549  3.39       0.66

      alcohol  quality
0         9.4        5
1         9.8        5
2         9.8        5
3         9.8        6
4         9.4        5
...       ...      ...
1594     10.5        5
1595     11.2        6
1596     11.0        6
1597     10.2        5
1598     11.0        6

[1599 rows x 12 columns]>
```

4.查看輸出(quality)

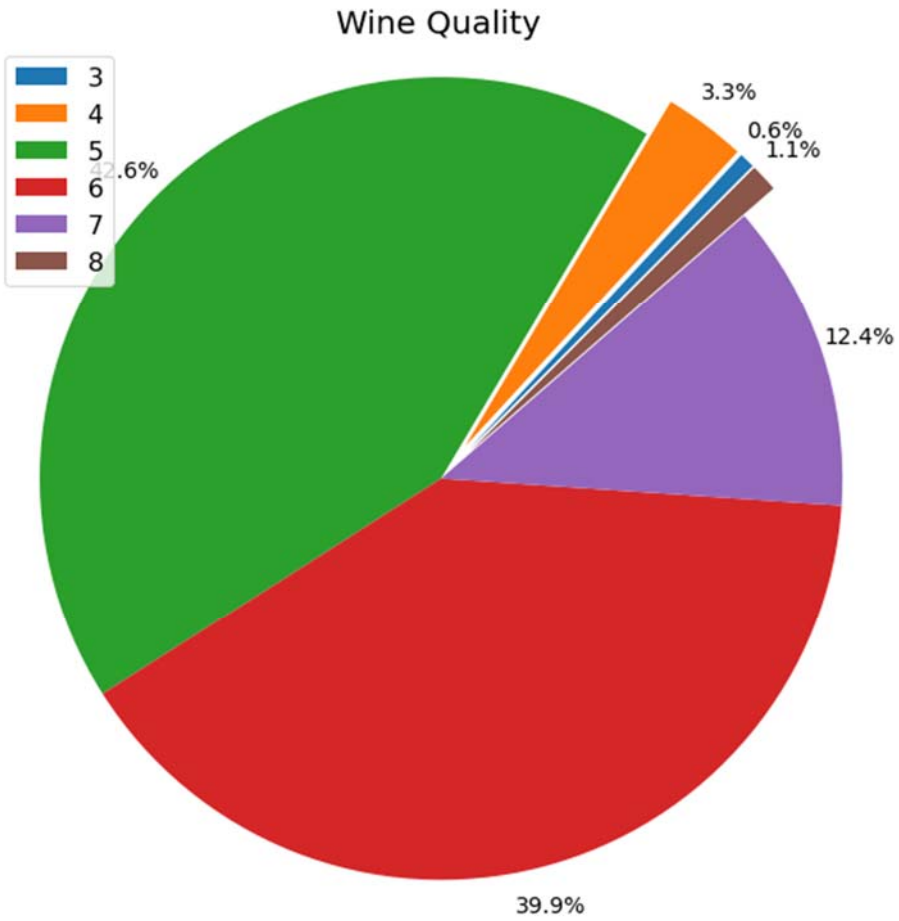
```
print("Quality Points:",df['quality'].unique())
```

```
Quality Points: [5 6 7 4 8 3]
```

```
print(df['quality'].value_counts())
```

```
5    681
6    638
7    199
4     53
8     18
3     10
Name: quality, dtype: int64
```

```
quality = df["quality"].value_counts().sort_index(ascending=True)
explode_list = [0.1, 0.1, 0, 0, 0, 0.1]
ax = quality.plot(kind='pie',
                  figsize = (10,10),
                  autopct='%1.1f%%',
                  startangle=45,
                  labels=None,
                  pctdistance=1.1,
                  explode=explode_list,
                  textprops={'fontsize': 14})
ax.patch.set_facecolor('white')
plt.title('Wine Quality', size = 20)
ax.set(ylabel=None)
plt.axis('equal')
plt.legend(labels=quality.index, loc="upper left", fontsize=16);
```



5.查看各項統計數值

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.319637	1.741096	4.60000	7.1000	7.90000	9.200000	15.90000
volatile acidity	1599.0	0.527821	0.179060	0.12000	0.3900	0.52000	0.640000	1.58000
citric acid	1599.0	0.270976	0.194801	0.00000	0.0900	0.26000	0.420000	1.00000
residual sugar	1599.0	2.538806	1.409928	0.90000	1.9000	2.20000	2.600000	15.50000
chlorides	1599.0	0.087467	0.047065	0.01200	0.0700	0.07900	0.090000	0.61100
free sulfur dioxide	1599.0	15.874922	10.460157	1.00000	7.0000	14.00000	21.000000	72.00000
total sulfur dioxide	1599.0	46.467792	32.895324	6.00000	22.0000	38.00000	62.000000	289.00000
density	1599.0	0.996747	0.001887	0.99007	0.9956	0.99675	0.997835	1.00369
pH	1599.0	3.311113	0.154386	2.74000	3.2100	3.31000	3.400000	4.01000
sulphates	1599.0	0.658149	0.169507	0.33000	0.5500	0.62000	0.730000	2.00000
alcohol	1599.0	10.422983	1.065668	8.40000	9.5000	10.20000	11.100000	14.90000
quality	1599.0	5.636023	0.807569	3.00000	5.0000	6.00000	6.000000	8.00000

6. 標記數據

```
features = ['residual sugar', 'total sulfur dioxide', 'sulphates',
                  'alcohol', 'volatile acidity', 'quality']
df_features = df[features]
```

```
case_1 = (df_features['quality']==3)|(df_features['quality']==4)
case_2 = (df_features['quality']==5)|(df_features['quality']==6)
case_3 = (df_features['quality']==7)|(df_features['quality']==8)
level_34 = round(df_features[case_1].describe(),2)
level_56 = round(df_features[case_2].describe(),2)
level_78 = round(df_features[case_3].describe(),2)
```

```
level_all = pd.concat([level_34,level_56, level_78],
                      axis=1,
                      keys=['3,4','5,6','7,8',])
level_all.T
```

		count	mean	std	min	25%	50%	75%	max
3,4	residual sugar	63.0	2.68	1.72	1.20	1.90	2.10	2.95	12.90
	total sulfur dioxide	63.0	34.44	26.40	7.00	13.50	26.00	48.00	119.00
	sulphates	63.0	0.59	0.22	0.33	0.50	0.56	0.60	2.00
	alcohol	63.0	10.22	0.92	8.40	9.60	10.00	11.00	13.10
	volatile acidity	63.0	0.72	0.25	0.23	0.56	0.68	0.88	1.58
	quality	63.0	3.84	0.37	3.00	4.00	4.00	4.00	4.00
5,6	residual sugar	1319.0	2.50	1.40	0.90	1.90	2.20	2.60	15.50
	total sulfur dioxide	1319.0	48.95	32.71	6.00	24.00	40.00	65.00	165.00
	sulphates	1319.0	0.65	0.17	0.37	0.54	0.61	0.70	1.98
	alcohol	1319.0	10.25	0.97	8.40	9.50	10.00	10.90	14.90
	volatile acidity	1319.0	0.54	0.17	0.16	0.41	0.54	0.64	1.33
	quality	1319.0	5.48	0.50	5.00	5.00	5.00	6.00	6.00
7,8	residual sugar	217.0	2.71	1.36	1.20	2.00	2.30	2.70	8.90
	total sulfur dioxide	217.0	34.89	32.57	7.00	17.00	27.00	43.00	289.00
	sulphates	217.0	0.74	0.13	0.39	0.65	0.74	0.82	1.36
	alcohol	217.0	11.52	1.00	9.20	10.80	11.60	12.20	14.00
	volatile acidity	217.0	0.41	0.14	0.12	0.30	0.37	0.49	0.92
	quality	217.0	7.08	0.28	7.00	7.00	7.00	7.00	8.00

7. 訓練機器學習模型

```
def get_models():
    models=[]
    models.append(("LR",LogisticRegression()))
    models.append(("NB",GaussianNB()))
    models.append(("KNN",KNeighborsClassifier()))
    models.append(("DT",DecisionTreeClassifier()))
    models.append(("SVM rbf",SVC()))
    models.append(("SVM linear",SVC(kernel='linear')))
    models.append(('LDA', LinearDiscriminantAnalysis()))

    return models

def cross_validation_scores_for_various_ml_models(X_cv, y_cv):
    print("success rates".upper())
    models=get_models()


    results=[]
    names= []

    for name, model in models:
        kfold=StratifiedKFold(n_splits=5,shuffle=True,random_state=22)
        cv_result=cross_val_score(model,X_cv, y_cv, cv=kfold,scoring="accuracy")
        names.append(name)
        results.append(cv_result)
        print("{} success rate:{:0.2f}".format(name, cv_result.mean()))

df_temp = df.copy(deep=True)
X = df.drop('quality', axis=1)
y = df['quality']

X=StandardScaler().fit_transform(X)
cross_validation_scores_for_various_ml_models(X, y)
```

```
SUCCESS RATES
LR success rate:nan
NB success rate:0.55
KNN success rate:0.58
DT success rate:0.61
SVM rbf success rate:0.62
SVM linear success rate:0.58
LDA success rate:0.59
```

```
df_temp = df.copy(deep=True)
X = df.drop('quality', axis=1)
y = df['quality']

X=StandardScaler().fit_transform(X)
cross_validation_scores_for_various_ml_models(X, y)
```

```
SUCCESS RATES
LR success rate:nan
NB success rate:0.55
KNN success rate:0.58
DT success rate:0.62
SVM rbf success rate:0.62
SVM linear success rate:0.58
LDA success rate:0.59
```

（五）My Insight：

這兩份 code 包含了視覺化的圖表、以及不同的機器學習模型，人們可以依照這些圖表去對葡萄酒進行評判參考，通過 python 程式碼，我們可以將數據的各項統計數值顯示出來，如五數摘要，透過機器學習，加上 cross validation 提高模型準確性，可以藉由葡萄酒的物理和化學性質，預測葡萄酒的品質，由此看出這份 Dataset 的價值。

參考文獻：

[1]：P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.