kaggle Search Competitions Datasets Notebooks Discussion Courses

- *Source:* UCI Machine Learning Repository
- *Input variables:*
  1 - fixed acidity
  2 - volatile acidity
  3 - citric acid
  4 - residual sugar
  5 - chlorides
  6 - free sulfur dioxide
  7 - total sulfur dioxide
  8 - density
  9 - pH
  10 - sulphates
  11 - alcohol
- *Output variable:* quality (score between 0 and 10)
- *Data Set Characteristics:* Multivariate
- *Number of Observations:* 1599
- *Number of Attributes/Variables:* 12
- *Missing Values:* N/A

Source: 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.

Setting up the development environment by importing required libraries and modules:

- *Numpy:* It will provide the support for efficient numerical computation.
- *Pandas:* It is convenient library that supports dataframes. Working with pandas will bring ease in many crucial data operations.
- *Matplotlib:* It provides a MATLAB-like plotting framework.
- *Seaborn:* It is a visualization library based on matplotlib which provides a high-level interface for drawing attractive statistical graphics.
- *Bokeh:* It is a interactive visualization library that targets modern web browsers for presentation.
- *Statsmodel:* It provides functions and classes for statistical tests and models.
- *Sklearn:* It is python library for data mining, data analysis and machine learning.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from bokeh.plotting import figure, output_file, show
from bokeh.layouts import row
from bokeh.io import output_notebook
import statsmodels.api as sm
import statsmodels.formula.api as smf
from patsy import dmatrices
import sklearn
import sklearn.metrics
from sklearn import ensemble
from sklearn import linear_model
```

```
import warnings
```

Notebook Data Comments

```
/opt/conda/lib/python3.6/site-packages/
statsmodels/compat/pandas.py:56: Future
Warning: The pandas.core.datetools modu
le is deprecated and will be removed in
a future version. Please use the panda
s.tseries module instead.
  from pandas.core import datetools
```

(https://bokeh.pydata.org) BokehJS 0.12.16 successfully loaded.

Loading the Red Wine dataset

- Lets read the red wine data set from the *'UCI Machine Learning Repository'*.
- Here, we can use the *read_csv()* from the *pandas* library to load data into dataframe from the remote url.

In [2]:

Output

```
url = "../input/winequality-red.csv"
wine = pd.read_csv(url)
```

- The *head(..)* function of *pandas* helps in viewing the preview of the dataset for n-number of rows

In [3]:

```
wine.head(n=5)
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	t s c
0	7.4	0.70	0.00	1.9	0.076	11.0	3
1	7.8	0.88	0.00	2.6	0.098	25.0	6
2	7.8	0.76	0.04	2.3	0.092	15.0	5
3	11.2	0.28	0.56	1.9	0.075	17.0	6
4	7.4	0.70	0.00	1.9	0.076	11.0	3

Exploring the Red Wine dataset:

In [4]:

```
print("Shape of Red Wine dataset: {s}".format(s = wi
ne.shape))
print("Column headers/names: {s}".format(s = list(wi
ne)))
```

```
Shape of Red Wine dataset: (1599, 12)
Column headers/names: ['fixed acidity',
'volatile acidity', 'citric acid', 'res
idual sugar', 'chlorides', 'free sulfur
dioxide', 'total sulfur dioxide', 'dens
ity', 'pH', 'sulphates', 'alcohol', 'qu
ality']
```

- From above lines we can learn that there are total *1599 observations with 12 different feature variables/attributes* present in the Red Wine dataset.

In [5]:

```
# Now, let's check the information about different va
riables/column from the dataset:
wine.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
fixed acidity           1599 non-null f
loat64
volatile acidity        1599 non-null f
loat64
citric acid             1599 non-null f
loat64
residual sugar          1599 non-null f
loat64
chlorides               1599 non-null f
loat64
free sulfur dioxide     1599 non-null f
loat64
total sulfur dioxide    1599 non-null f
loat64
density                 1599 non-null f
loat64
```

```
loat64
pH                       1599 non-null f
loat64
sulphates                1599 non-null f
loat64
alcohol                  1599 non-null f
loat64
quality                  1599 non-null i
nt64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

- We can see that, all 12 columns are of numeric data types. Out of 12 variables, 11 are predictor variables and last one *'quality'* is an response variable.

In [6]:

```
# Let's look at the summary of the dataset,
wine.describe()
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar
count	1599.000000	1599.000000	1599.000000	1599.000
mean	8.319637	0.527821	0.270976	2.538806
std	1.741096	0.179060	0.194801	1.409928
min	4.600000	0.120000	0.000000	0.900000
25%	7.100000	0.390000	0.090000	1.900000
50%	7.900000	0.520000	0.260000	2.200000
75%	9.200000	0.640000	0.420000	2.600000
max	15.900000	1.580000	1.000000	15.50000

- The summary of Red Wine dataset looks perfect, there is no visible abnormality in data (invalid/negative values).
- All the data seems to be in range (with different scales, which needs standardization).

- Let's look for the missing values in red wine dataset:

In [7]:

```
wine.isnull().sum()
```

Out[7]:

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

- The red wine dataset doesn't have any missing values/rows/cells for any of the variables/feature.
- It seems that data has been collected neatly or prior cleaning has been performed before publishing the dataset.

- Let's rename the modify the dataset headers/column names by removing the *'blank spaces'* from it.

In [8]:

```
wine.rename(columns={'fixed acidity': 'fixed_acidit
y','citric acid':'citric_acid','volatile acidity':'v
olatile_acidity','residual sugar':'residual_sugar',
'free sulfur dioxide':'free_sulfur_dioxide','total s
ulfur dioxide':'total_sulfur_dioxide'}, inplace=True
)
wine.head(n=5)
```

Out[8]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar
0	7.4	0.70	0.00	1.9
1	7.8	0.88	0.00	2.6
2	7.8	0.76	0.04	2.3
3	11.2	0.28	0.56	1.9
4	7.4	0.70	0.00	1.9

Learning more about the target/response variable/feature:

- Let's check how many unique values does the target feature *'quality'* has?

In [9]:

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

Out[9]:

```
array([5, 6, 7, 4, 8, 3])
```

- And how data is distributed among those values?

In [10]:

```
wine.quality.value_counts().sort_index()
```

Out[10]:

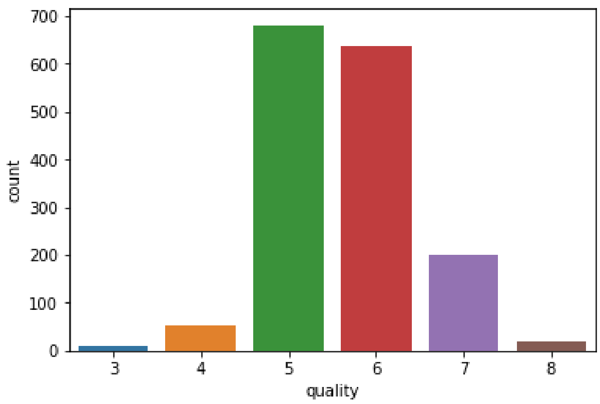
```
3     10
4     53
5    681
6    638
7    199
8     18
Name: quality, dtype: int64
```

In [11]:

```
sns.countplot(x='quality', data=wine)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot
at 0x7faab5e012b0>
```



- The above distribution shows the range for response variable (*quality*) is between 3 to 8.

- Let's create a new discreet, categorical response variable/feature ('*rating*') from existing '*quality*' variable.
i.e. bad: 1-4

i.e. bad: 1-4
average: 5-6
good: 7-10

In [12]:

```
conditions = [
    (wine['quality'] >= 7),
    (wine['quality'] <= 4)
]
rating = ['good', 'bad']
wine['rating'] = np.select(conditions, rating, defau
lt='average')
wine.rating.value_counts()
```

Out[12]:

```
average    1319
good        217
bad          63
Name: rating, dtype: int64
```

In [13]:

```
wine.groupby('rating').mean()
```

Out[13]:

	fixed_acidity	volatile_acidity	citric_acid	residual_
rating				
average	8.254284	0.538560	0.258264	2.503867
bad	7.871429	0.724206	0.173651	2.684921
good	8.847005	0.405530	0.376498	2.70875€

Corelation between features/variables:

- Let's check the corelation between the target variable and predictor variables,

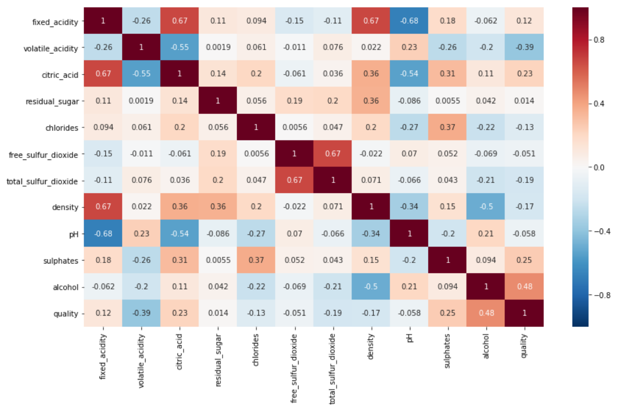
In [14]:

```
correlation = wine.corr()
plt.figure(figsize=(14, 8))
sns.heatmap(correlation, annot=True, linewidths=0, v
min=-1, cmap="RdBu_r")
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot
```

```
at 0x7faab2490d68>
```



In [15]:

```
correlation['quality'].sort_values(ascending=False)
```

Out[15]:

```
quality                 1.000000
alcohol                 0.476166
sulphates               0.251397
citric_acid             0.226373
fixed_acidity           0.124052
residual_sugar          0.013732
free_sulfur_dioxide    -0.050656
pH                     -0.057731
chlorides              -0.128907
density                -0.174919
total_sulfur_dioxide   -0.185100
volatile_acidity       -0.390558
Name: quality, dtype: float64
```

- We can observe that, the *'alcohol, sulphates, citric_acid & fixed_acidity'* have maximum corelation with response variable '*quality*'.
- This means that, they need to be further analysed for detailed pattern and corelation exploration. Hence, we will use only these 4 variables in our future analysis.

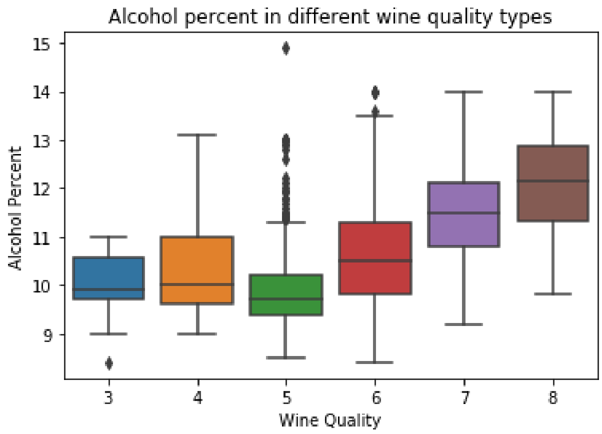
Analysis of alcohol percentage with wine quality:

In [16]:

```
bx = sns.boxplot(x="quality", y='alcohol', data = wi
ne)
bx.set(xlabel='Wine Quality', ylabel='Alcohol Percen
t', title='Alcohol percent in different wine quality
types')
```

Out[16]:

```
[Text(0,0.5,'Alcohol Percent'),
 Text(0.5,0,'Wine Quality'),
 Text(0.5,1,'Alcohol percent in differe
nt wine quality types')]
```



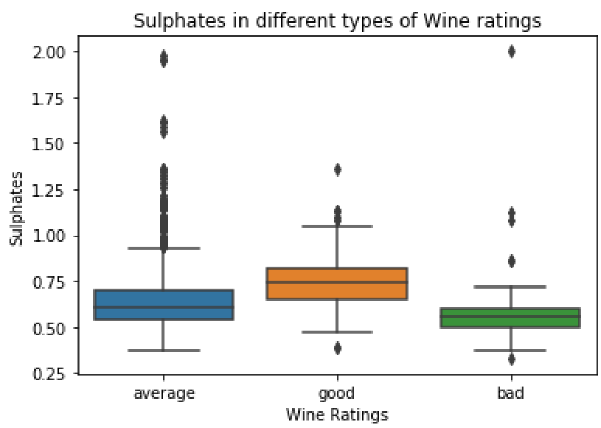
Analysis of sulphates & wine ratings:

In [17]:

```
bx = sns.boxplot(x="rating", y='sulphates', data = w
ine)
bx.set(xlabel='Wine Ratings', ylabel='Sulphates', ti
tle='Sulphates in different types of Wine ratings')
```

Out[17]:

```
[Text(0,0.5,'Sulphates'),
 Text(0.5,0,'Wine Ratings'),
 Text(0.5,1,'Sulphates in different typ
es of Wine ratings')]
```



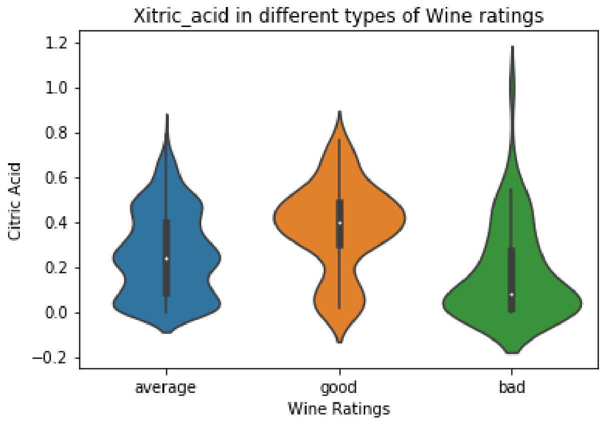
Analysis of Citric Acid & wine ratings:

In [18]:

```
bx = sns.violinplot(x="rating", y='citric_acid', dat
a = wine)
bx.set(xlabel='Wine Ratings', ylabel='Citric Acid',
title='Xitric_acid in different types of Wine rating
s')
```

Out[18]:

```
[Text(0,0.5,'Citric Acid'),
 Text(0.5,0,'Wine Ratings'),
 Text(0.5,1,'Xitric_acid in different t
ypes of Wine ratings')]
```



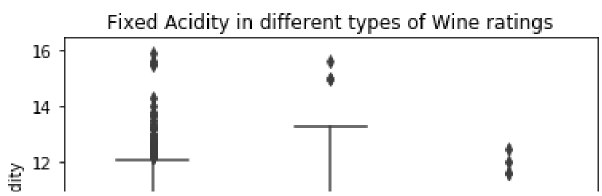
Analysis of fixed acidity & wine ratings:

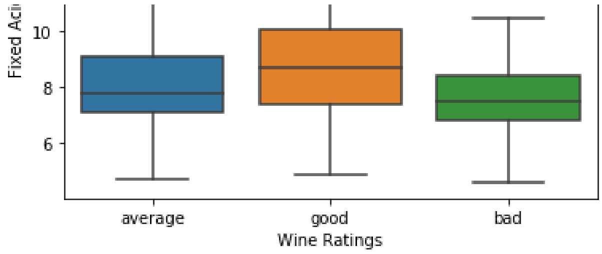
In [19]:

```
bx = sns.boxplot(x="rating", y='fixed_acidity', data
= wine)
bx.set(xlabel='Wine Ratings', ylabel='Fixed Acidity'
, title='Fixed Acidity in different types of Wine ra
tings')
```

Out[19]:

```
[Text(0,0.5,'Fixed Acidity'),
 Text(0.5,0,'Wine Ratings'),
 Text(0.5,1,'Fixed Acidity in different
types of Wine ratings')]
```





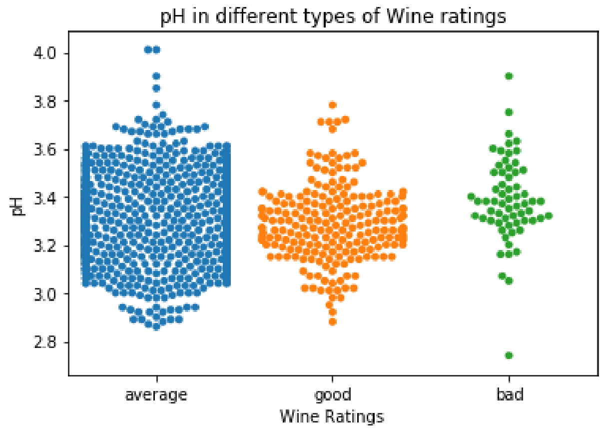
Analysis of pH & wine ratings:

In [20]:

```
bx = sns.swarmplot(x="rating", y="pH", data = wine);
bx.set(xlabel='Wine Ratings', ylabel='pH', title='pH
in different types of Wine ratings')
```

Out[20]:

```
[Text(0,0.5,'pH'),
 Text(0.5,0,'Wine Ratings'),
 Text(0.5,1,'pH in different types of W
ine ratings')]
```



Linear Regression:

- Below graphs for different quality ratings shows a linear regression between residual_sugar & alcohol in red wine,

In [21]:

```
sns.lmplot(x = "alcohol", y = "residual_sugar", col
= "rating", data = wine)
```

Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0x7faaac
```

```
5b33c8>
```

- The linear regression plots above for different wine quality ratings (bad, average & good) shows the regression between alcohol and residual sugar content of the red wine.
- We can observe from the trendline that, for good and average wine types the residual sugar content remains almost constant irrespective of alcohol content value. Whereas for bad quality wine, the residual sugar content increases gradually with the increase in alcohol content.
- This analysis can help in manufacturing the good quality wine with continuous monitoring and contrilling the alcohol and residual sugar content of the red wine.

In [22]:

```
y,X = dmatrices('quality ~ alcohol', data=wine, retu
rn_type='dataframe')
print("X:", type(X))
print(X.columns)
model=smf.OLS(y, X)
result=model.fit()
result.summary()
```

```
X: <class 'pandas.core.frame.DataFram
e'>
Index(['Intercept', 'alcohol'], dtype
='object')
```

Out[22]:

OLS Regression Results

Dep. Variable:	quality	R-squared:	0.227
Model:	OLS	Adj. R-squared:	0.226
Method:	Least Squares	F-statistic:	468.3
Date:	Sat, 26 May 2018	Prob (F-statistic):	2.83e-91
Time:	22:53:46	Log-Likelihood:	-1721.1
No. Observations:	1599	AIC:	3446.
Df Residuals:	1597	BIC:	3457.
Df Model:	1		

Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.8750	0.175	10.732	0.000	1.532	2.218
alcohol	0.3608	0.017	21.639	0.000	0.328	0.394

Omnibus:	38.501	Durbin-Watson:	1.748
Prob(Omnibus):	0.000	Jarque-Bera (JB):	71.758
Skew:	-0.154	Prob(JB):	2.62e-16
Kurtosis:	3.991	Cond. No.	104.

In [23]:

```
model = smf.OLS.from_formula('quality ~ alcohol', da
ta = wine)
results = model.fit()
print(results.params)
```

```
Intercept    1.874975
alcohol      0.360842
dtype: float64
```

- The above wine quality vs alcohol content regression model's result shows that, the minimum value for quality is 1.87 and there will be increment by single unit for wine quality for every change of 0.360842 alcohol units.

Classification

Classification using Statsmodel:

- We will use statsmodel for this logistic regression analysis of predicting good wine quality (>4).
- Let's create a new categorical variable/column (rate_code) with two possible values (good = 1 & bad = 0).

In [24]:

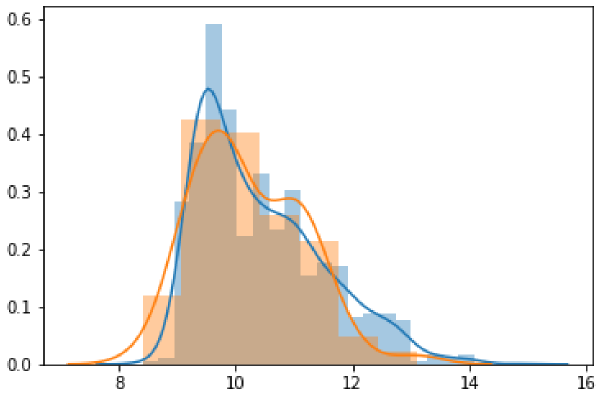
```
wine['rate_code'] = (wine['quality'] > 4).astype(np.
float32)
```

In [25]:

```
y, X = dmatrices('rate_code ~ alcohol', data = wine)
sns.distplot(X[y[:,0] > 0, 1])
sns.distplot(X[y[:,0] == 0, 1])
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot
at 0x7faaa433ffd0>
```



- The above plot shows the higher probability for red wine quality will be good if alcohol percentage is more than equal to 12, whereas the same probability reduces as alcohol percentage decreases.

In [26]:

```
model = smf.Logit(y, X)
result = model.fit()
result.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.1652
09
         Iterations 8
```

Out[26]:

Model:	Logit	No. Iterations:	8.0000
Dependent Variable:	rate_code	Pseudo R-squared:	0.005
Date:	2018-05-26 22:53	AIC:	532.3386
No. Observations:	1599	BIC:	543.0928
Df Model:	1	Log-Likelihood:	-264.17
Df Residuals:	1597	LL-Null:	-265.48
Converged:	1.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.9
Intercept	1.0456	1.3628	0.7673	0.4429	-1.6253	3.7
alcohol	0.2082	0.1327	1.5685	0.1168	-0.0519	0.4

Did you find this Kernel useful?
Show your appreciation with an upvote

5

Data

Data Sources

Red Wine Quality
winequality-red.csv 1599 x 12

Red Wine Quality

Simple and clean practice dataset for regression or classification modelling

Last Updated: 2 years ago (Version 2)

About this Dataset

Context

The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. For more details, consult the reference [Cortez et al., 2009]. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

These datasets can be viewed as classification or regression tasks. The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones).

This dataset is also available from the UCI machine learning repository, https://archive.ics.uci.edu/ml/datasets/wine+quality , I just shared it to kaggle for convenience. (If I am mistaken and the public license type disallowed me from doing so, I will take this down if requested.)

Content

For more information, read [Cortez et al., 2009].
Input variables (based on physicochemical tests):
1 - fixed acidity
2 - volatile acidity

Comments (0)

Click here to comment...