Support Vector Machine Technique

Support Vector Machines are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.

In [1]:

```
#Importing the libraries "pandas, numpy, matplotlib.pyplot and seaborn" to my python scri
pt
#with the standard short name as "pd, np, plt and sns".

#Uploading the file on google colab and choosing the selected dataset by clicking "choose
files".

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
%matplotlib inline
from google.colab import files
uploaded = files.upload()
```

Choose File **No file selected**

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving New_Data3.csv to New_Data3.csv
```

Reading the data

In [2]:

```
#In the first step, importing the dataset in the project by the "read_csv" function
#and that reads the data into a pandas dataframe object.

datadf = pd.read_csv('New_Data3.csv')
datadf.head()
```

Out[2]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour	Grade
0	6.6	35	1	0	1	0	254	high
1	6.6	36	0	1	0	1	253	high
2	8.5	70	1	1	1	1	246	low
3	9.5	34	1	1	0	1	255	low
4	6.6	37	0	0	0	0	255	medium

Analyzing the data

In [3]:

```
#The shape is a tuple of the array dimensions which gives the number of rows and
#columns of a given dataframe.

datadf.shape
```

Out[3]:

```
(1059, 8)
```

In [4]:

```
#The info() method gives the information about the dataframe where the information contai
ns;
#number of columns
#column labels
#column data types
#memory usage
#range index
#number of cells in each column

datadf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1059 entries, 0 to 1058
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   pH          1059 non-null   float64
 1   Temprature  1059 non-null   int64  
 2   Taste       1059 non-null   int64  
 3   Odor        1059 non-null   int64  
 4   Fat         1059 non-null   int64  
 5   Turbidity   1059 non-null   int64  
 6   Colour      1059 non-null   int64  
 7   Grade       1059 non-null   object 
dtypes: float64(1), int64(6), object(1)
memory usage: 66.3+ KB
```

In [5]:

```
#The describe() method gives description of the data in the dataframe.
#The details are included in the description for each column if the dataframe includes nu
merical data.

datadf.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
pH	1059.0	6.630123	1.399679	3.0	6.5	6.7	6.8	9.5
Temprature	1059.0	44.226629	10.098364	34.0	38.0	41.0	45.0	90.0
Taste	1059.0	0.546742	0.498046	0.0	0.0	1.0	1.0	1.0
Odor	1059.0	0.432483	0.495655	0.0	0.0	0.0	1.0	1.0
Fat	1059.0	0.671388	0.469930	0.0	0.0	1.0	1.0	1.0
Turbidity	1059.0	0.491029	0.500156	0.0	0.0	0.0	1.0	1.0
Colour	1059.0	251.840415	4.307424	240.0	250.0	255.0	255.0	255.0

Data Cleaning

Analyzing the numerical and categorical features, and convert categorical feature into numerical.

In [6]:

```
#In this unique() function, we have to pass the series as a parameter to find
#the unique values of an array and these values as a sorted array.

datadf['Grade'].unique()
```

Out[6]:

```
array(['high', 'low', 'medium'], dtype=object)
```

Check for missing values

In [7]:

In [7]:

```
#The function isnull().sum() returns the number of missing values in the dataset.

datadf.isnull().sum()
```

Out[7]:

```
pH            0
Temprature    0
Taste         0
Odor          0
Fat           0
Turbidity     0
Colour        0
Grade         0
dtype: int64
```

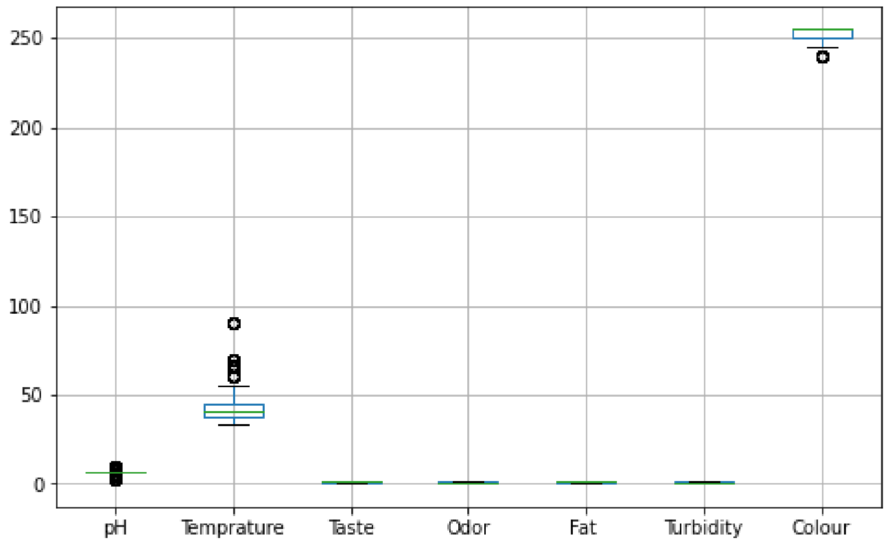
EDA - EXPLORATORY DATA ANALYSIS

In [8]:

```
#Checking for outliers.
#Non-graphical format

#A boxplot is a common visual representation of data acquisition based on a digit summary
.
#It can inform of the amounts of the outliers.

plt.figure(figsize=(8,5))
datadf.boxplot()
```

Out[8]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ff0f4fc3c10>



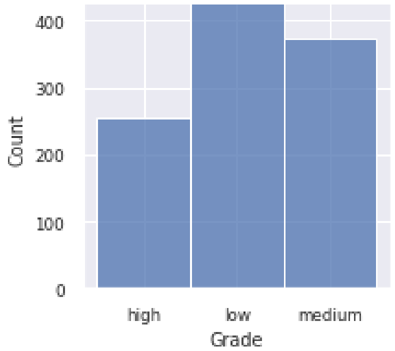
Graphical

In [9]:

```
#Graphical format.

#Histplot function is a traditional visualization tool that counts the number of data
#that fall into discrete bins to illustrate the distribution of one or more variables.

plt.figure(figsize=(3,3))
sns.set(font_scale=0.8)
sns.histplot(data=datadf, x='Grade')
```

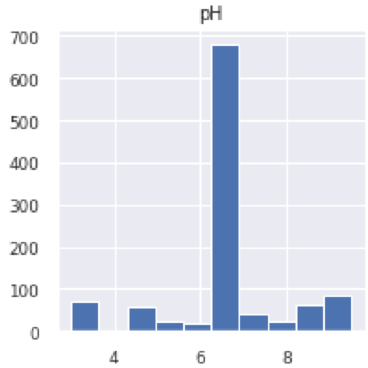
Out[9]:

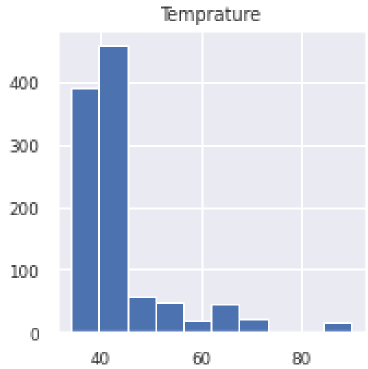
<matplotlib.axes._subplots.AxesSubplot at 0x7ff0bdcf08b0>

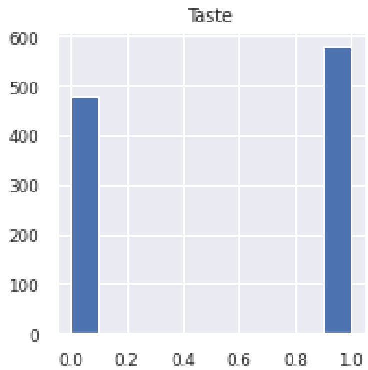


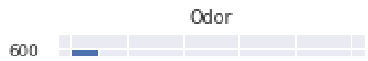
In [10]:

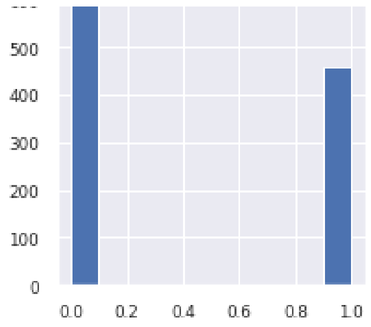
```
#The below function hist() showing all the categories in column with different plot.
#plt.show() function shows all the columns in separate format.
#pH
#Temperature
#Taste
#Odor
#Turbidity
#Colour

column=['pH','Temprature','Taste','Odor','Turbidity','Colour']
for category in column:
    plt.figure(figsize=(3,3))
    plt.hist(datadf[category])
    plt.title(category)
    plt.show()
```

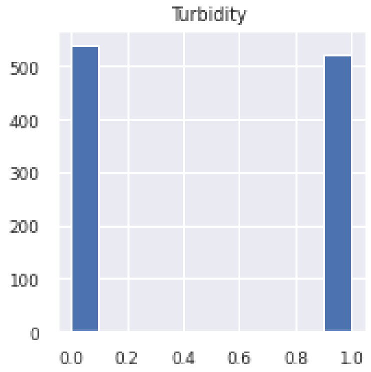


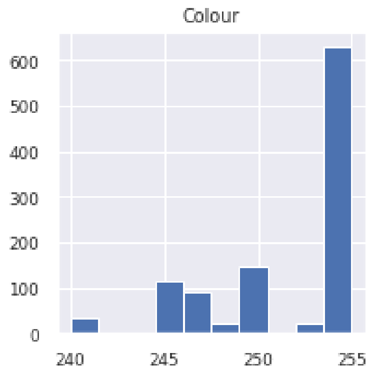












Correlation

In [11]:

```
# corr() function is used to find the correlation among the columns in the Dataframe

correlation = datadf.corr()
correlation
```

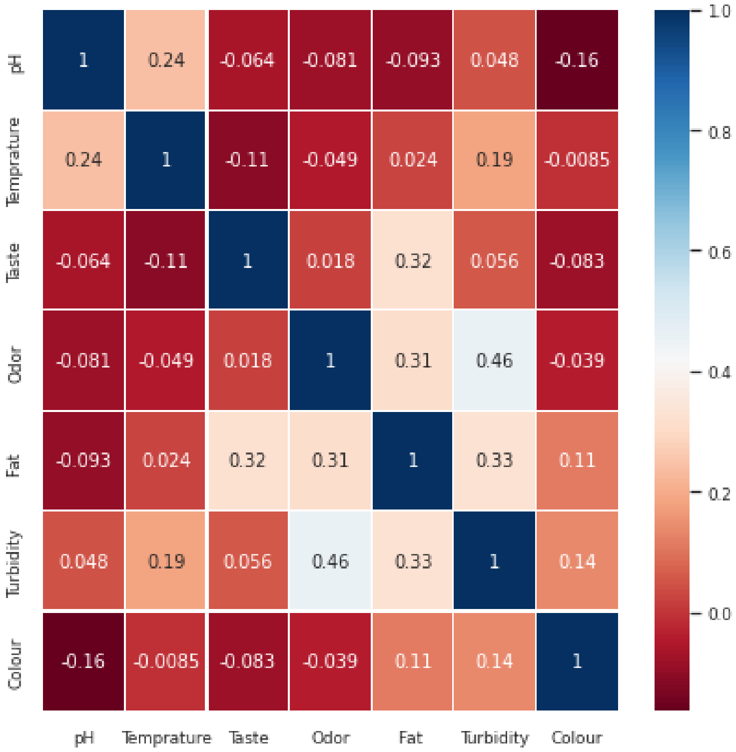
Out[11]:

	pH	Temprature	Taste	Odor	Fat	Turbidity	Colour
pH	1.000000	0.244684	-0.064053	-0.081331	-0.093429	0.048384	-0.164565
Temprature	0.244684	1.000000	-0.109792	-0.048870	0.024073	0.185106	-0.008511
Taste	-0.064053	-0.109792	1.000000	0.017582	0.324149	0.055755	-0.082654
Odor	-0.081331	-0.048870	0.017582	1.000000	0.314505	0.457935	-0.039361
Fat	-0.093429	0.024073	0.324149	0.314505	1.000000	0.329264	0.114151
Turbidity	0.048384	0.185106	0.055755	0.457935	0.329264	1.000000	0.136436
Colour	-0.164565	-0.008511	-0.082654	-0.039361	0.114151	0.136436	1.000000

In [12]:

```
#sns.heatmap function is a great way to visualize data, because it can show the
#relation between variabels including time

#showing all the columns in the perfect figsize.

plt.figure(figsize=(7,7))
```

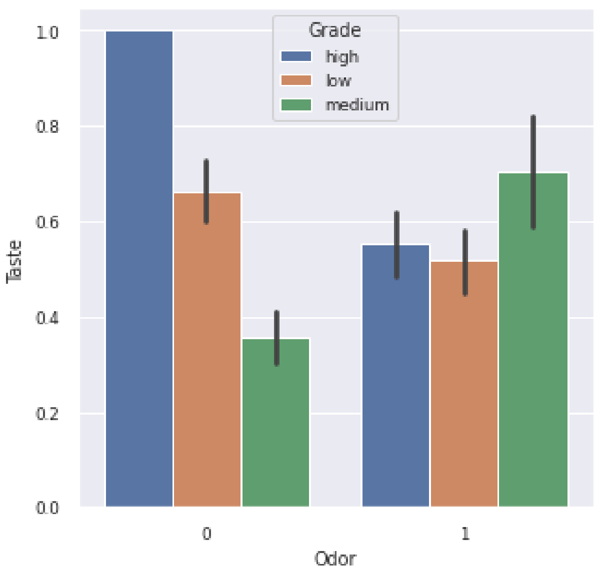
```
sns.heatmap(correlation,annot=True,cmap='RdBu',linewidths=0.2)
plt.show()
```



Multivariate Graphical

In [13]:

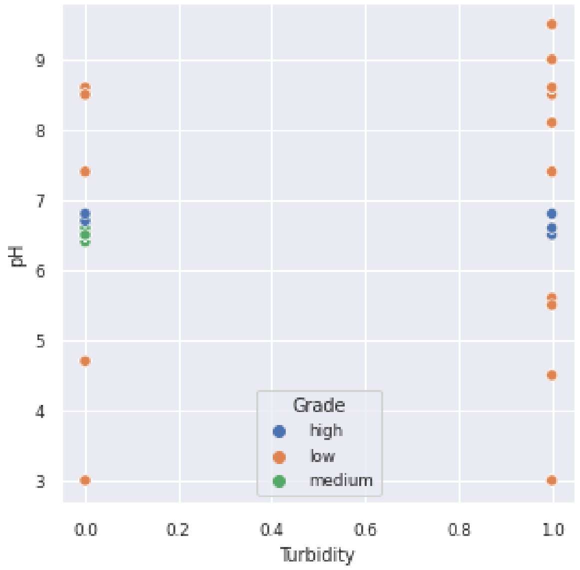
```
#A barplot() function represents an estimate of central tendency for a numeric
#variable with the height of each rectangle.

#showing odor in x axis and taste in y axis with three different grades as high, low and
medium.

plt.figure(figsize=(5,5))
sns.barplot(x='Odor',y='Taste',hue='Grade',data=datadf)
plt.show()
```



In [14]:

```
#The scatterplot() function displays data between two continuous data. It shows
#how one data variable affects the other variable.

#showing turbidity in x axis and pH in y axis with three different grades as high, low an
d medium.
```

```
plt.figure(figsize=(5,5))
sns.scatterplot(x='Turbidity',y='pH',hue='Grade',data=datadf)
plt.show()
```



In [15]:

```
#Importing preprocessing which provides several common utility functions and
#transformer classes to change raw feature vectors into a representation.

#Next preprocessing.LabelEncoder() which is used to encode labels with a value
#between 0 and n_classes-1 where n is the number of distinct labels.

from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
datadf['Grade']= label_encoder.fit_transform(datadf['Grade'])
datadf['Grade'].unique()
```

Out[15]:

```
array([0, 1, 2])
```

In [16]:

```
#Importing train_test_split function to split arrays or matrices into
#random subsets for train and test data.

#Splitting the data using train_test_split.
#Dropping the Grade in axis.

from sklearn.model_selection import train_test_split
X=datadf.drop(['Grade'],axis=1)
y=datadf['Grade']
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=2)
```

Support Vector Machine Technique

In [17]:

```
#Importing SVC model to predict the values.

#Importing accuracy_score which calculates the accuracy score for a set of
#predicted labels against the true labels.

#.fit() is used to evaluate how the classifier performs on the training set with .score (
X_train, Y_train).

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(X_train, y_train)
```

Out[17]:

```
SVC(kernel='linear', random_state=0)
```

In [18]:

```
#Here is the final prediction and the accuracy score.

y_predict= classifier.predict(X_test)
score=accuracy_score(y_test,y_predict)
print(score)
```

```
0.8427672955974843
```