M507-A-Method of Prediction Assessment -GH1019718

June 29, 2022

1 Wine Quality Prediction Assessment

[38]:
```
## Importing image using IPython.image code:
from IPython.display import Image
from IPython.core.display import HTML
Image(url= "https://crosser.io/media/wp3ngty3/edge-mlops-workflow.svg?
 ↪width=500&height=125")
```

[38]:
```
<IPython.core.display.Image object>
```

1.1 List of Content :

1.2 1. Introdution

I am working on a dataset which is called **winequality** which is produced by a wine producer company.Because satisfied consumers increase profit and revenue for businesses, many organizations place a high priority on customer satisfaction. A business can identify what aspects of its goods, services, and internal procedures are effective and what needs to be changed or improved by measuring customer satisfaction. Gaining knowledge of client satisfaction can help you inhance sales abilities and offer customers better goods and services.As a result of this the wine producer company request us to predict future customer satisfaction by using Machine Learning algorithms .The main aim and problem statement of this compnay is that future customers are going to be

satisfied with the company products or not ? In this Note Book I am Going to use Machine learning models and I will report the result to the senior manager for further decison making .

1.3 2. import requierd libraries

[37]:
```
## Importing image using IPython.image code:
from IPython.display import Image
from IPython.core.display import HTML
Image(url= "https://cdn.educba.com/academy/wp-content/uploads/2019/09/
 ↪Machine-Learning-Libraries-1.png")
```

[37]:
```
<IPython.core.display.Image object>
```

1. **Pandas** : Pandas, is an open-source python library that provides a variety of flexible, high-performance, and easy-to-use data structures for users for instance series and data frames. while Python is a useful language for data preparation but it has shortcomings in the efficiency when it comes to data analysis and modelling.
2. **numpy**: NumPy is one of the most basic data handling libraries used in Python-based scientific computing. The ability to conduct mathematical operations on a sizable N-dimensional array is provided by **Numpy**.
3. **Matplotlib**: The point I would like to mention about Matplotliblibraryy is that **Matplotlib** is a data visualization library that works with NumPy, pandas, and other interactive environments across platforms. Matplotlib can be used in a variety of data visualization tasks for example to plot charts, axis, figures, or publications, and it is easy to use in Jupiter notebooks
4. **Seaborn**: With fewer lines of code, the seaborn library provides attractive information by generating graphs in order to aggregate statistics, especially for categorical and multivariate data. Ref:https://www.educba.com/machine-learning-libraries

[5]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
#Importing required packages.
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
import matplotlib.pyplot as plt
```

1.4 3. loading the dataset

- I load my Data set using read.csv methode
- I call it **winequal**

[6]:
```
winequal = pd.read_csv("winequality-red.csv")
winequal.head()
```

[6]:

	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

I get the number and shape of datas by using the name of my DataFrame and shape function.

[7]:
```
print (winequal.shape)
```

```
(1599, 12)
```

1.5 4. Data Exploration

4.1. Info Getting the Information about my data by runnning **info** function and my dataframe name,winequal.These info includes The number of columns,labls of these columns,types of these columns , memory usage, range index of the dataset, and one more thing to be added is the number of cells in each column also known as non-null values.

[8]:
```
winequal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
```

```
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

4.2. Describe The code Describe() ,describe the datafram by explaining the counting number of not-empty datas,evaluating the mean,standard deviation and minimum of values

[9]:
```
winequal.describe()
```

[9]:

	fixed acidity	volatile acidity	citric acid	residual sugar	\
count	1599.000000	1599.000000	1599.000000	1599.000000	
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	15.500000	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	\
count	1599.000000	1599.000000	1599.000000	1599.000000	
mean	0.087467	15.874922	46.467792	0.996747	
std	0.047065	10.460157	32.895324	0.001887	
min	0.012000	1.000000	6.000000	0.990070	
25%	0.070000	7.000000	22.000000	0.995600	
50%	0.079000	14.000000	38.000000	0.996750	
75%	0.090000	21.000000	62.000000	0.997835	
max	0.611000	72.000000	289.000000	1.003690	

	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000
mean	3.311113	0.658149	10.422983	5.636023
std	0.154386	0.169507	1.065668	0.807569
min	2.740000	0.330000	8.400000	3.000000
25%	3.210000	0.550000	9.500000	5.000000

```
50%       3.310000      0.620000      10.200000      6.000000
75%       3.400000      0.730000      11.100000      6.000000
max       4.010000      2.000000      14.900000      8.000000
```

4.3. NULL Article I checked the empty or null values with this code,As it is apparent from the result there is no null value in my dataframe other wise I had to replcae this article with a methode to have better performance.

[10]:
```
# checking for missing values
winequal.isnull().sum()
```

[10]:
```
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
```

4.5. Counting the Quality value counting quality values

[11]:
```
winequal.quality.value_counts().sort_index()
```

[11]:
```
3     10
4     53
5    681
6    638
7    199
8     18
Name: quality, dtype: int64
```

4.6 Balancing the data In order to have better model performance I use this code to balance my dataset .

[12]:
```
winequal.replace({'quality':{3:0, 4:0, 5:0, 6:1, 7:1, 8:1 }}, inplace = True)
winequal['quality'].value_counts().sort_index()
```

[12]:
```
0    744
1    855
```

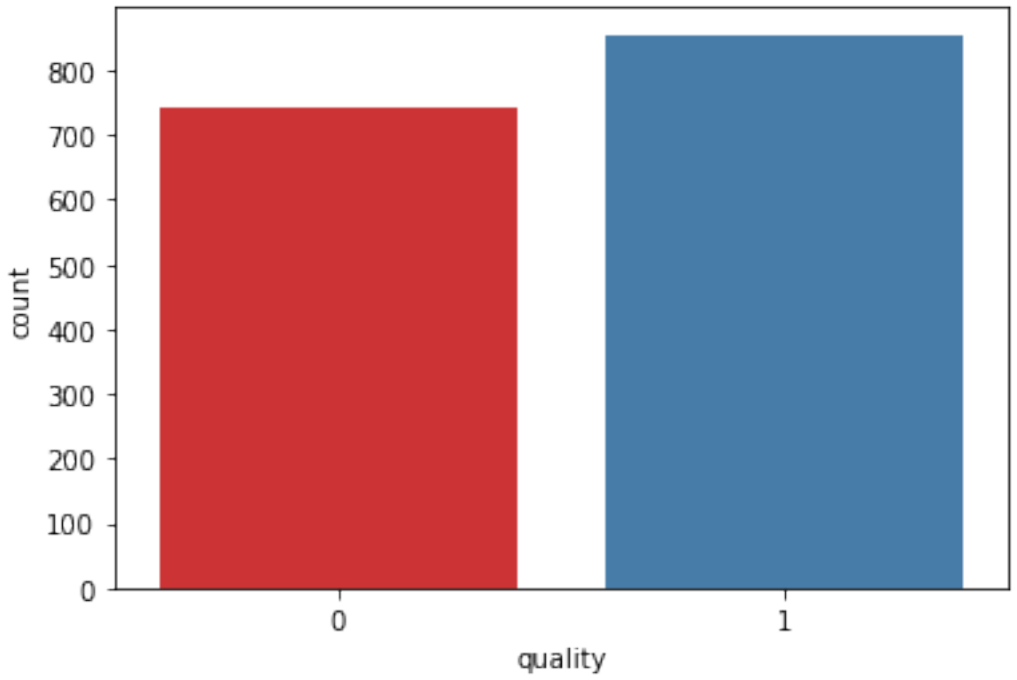
```
Name: quality, dtype: int64
```

1.6 5. Data Visualization

In my library I imported seaborn library in order to use **sns.countplot**, that can literaly counts the number of observations per category for a categorical variable, and displays the results as a bar chart. To be more specefic, the Seaborn countplot() aim to create a type of bar chart in Python.(https://www.sharpsightlabs.com/blog/seaborn-countplot/)

[13]:
```
sns.countplot(x='quality', data=winequal, palette="Set1")
```

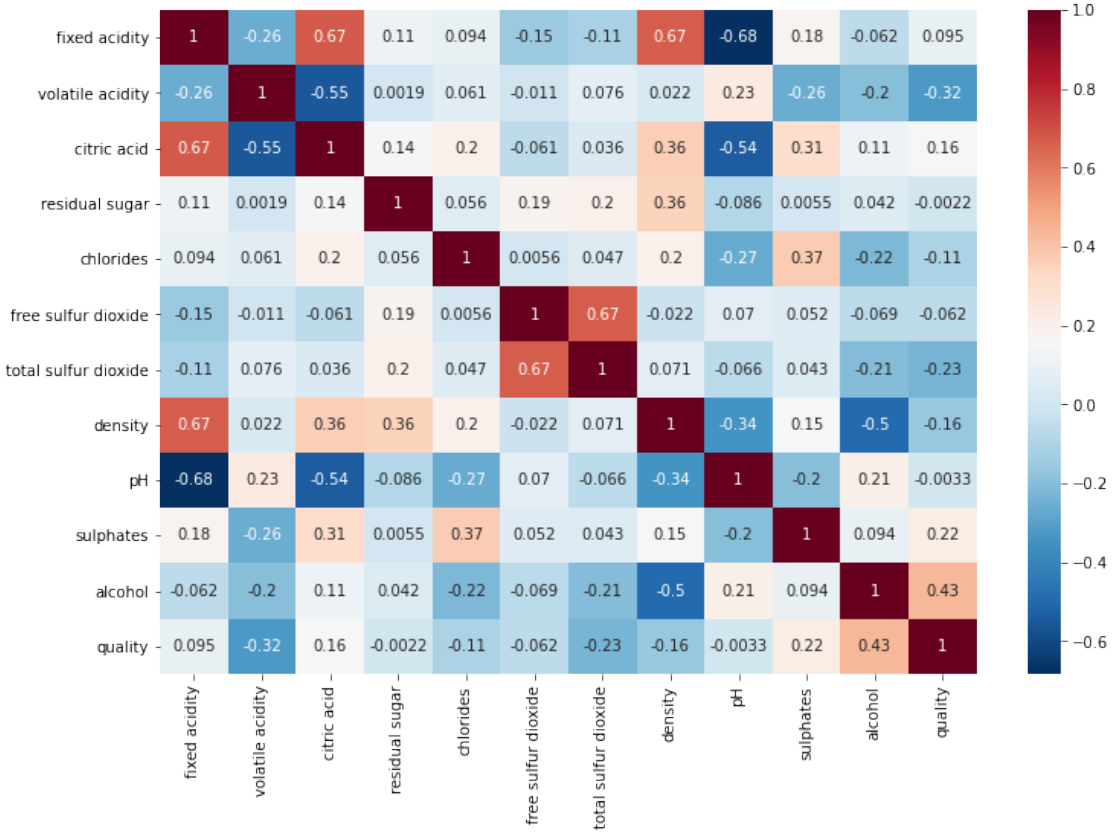
[13]:
```
<AxesSubplot:xlabel='quality', ylabel='count'>
```



In Order to check the correlation and realationship of columns i used seaborn and sns.heatmap function to display it in a 2D dimensional graphical format.

[14]:
```
# looking for correlation between columns
plt.figure(figsize=(12,8))
sns.heatmap(winequal.corr(), annot=True, cmap = 'RdBu_r')
```

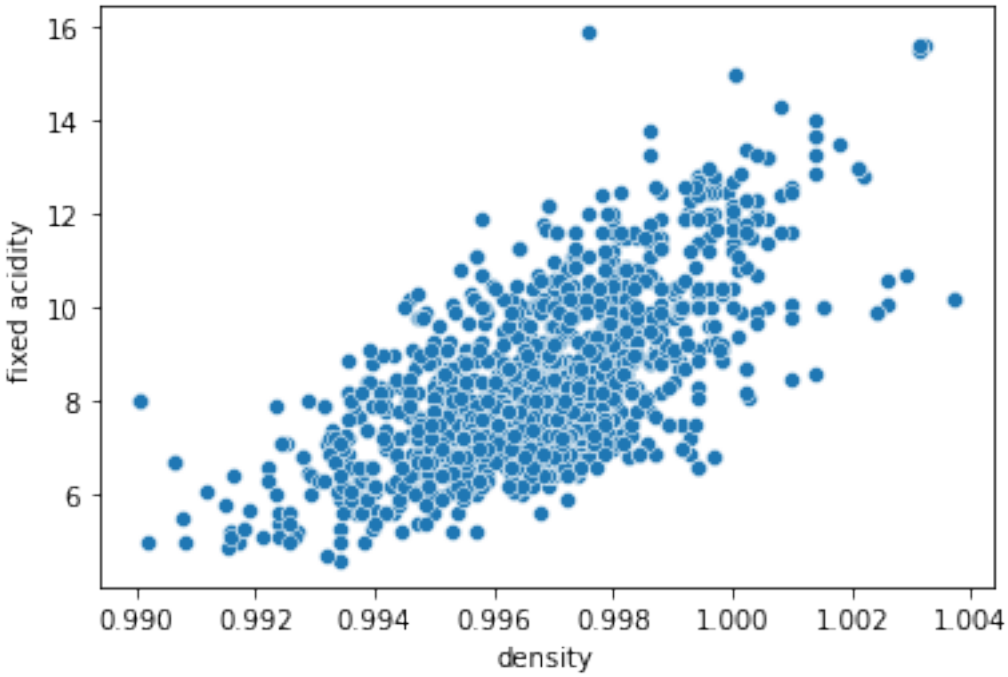
[14]:
```
<AxesSubplot:>
```



I Tried to display association between variables and its density by sns.scatterplot function and importing sns library.

[15]:
```
sns.scatterplot(x='density', y='fixed acidity', data=winequal)
```

[15]:
```
<AxesSubplot:xlabel='density', ylabel='fixed acidity'>
```



1.7 6. Data preprocessing

I seprate the data set into response variable and feature variabes.

[16]:
```
features = winequal.drop(['quality'] , axis = 1)
target = winequal['quality']
```

[17]:
```
X = winequal.drop('quality', axis = 1)
y = winequal['quality']
```

1.8 7. Train and Test splitting

In the following step I would like to split my data set into test and train splitt.To shed light on the process I put the test size into 0.2 which means I gave 20% of my total number of dataset into test dataset and random state into 42 which means with this parameter I tried to to fix the way the data is being sampled each time.As a result of this if we want to reproduce the same model again we will get the same result not a new result each time.

[18]:
```
#Train and Test splitting of data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 42)
```

[19]:
```
#Applying Standard scaling to get optimized result
sc = StandardScaler()
```

[20]:
```
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Our training and testing data is ready now to perform machine learning algorithm.

1.9 8. Machin Learning Models

1.9.1 8.1. Random Forest Classifier

Random Forest is a supervised learning algorithm which is used not only for classification but also for regression. The next point I would like to mention here is that random forest consist of many trees and this forest is stronger in comparison to each of Trees. A forest is expected to be stronger due to the more trees it has. On randomly chosen data samples, random forests make decision trees, do the predictions from each tree, and choose best one. (https://www.datacamp.com/tutorial/random-foreAdditionallysts-classifier-python) I modify RandomForestClassifier as a **rfcWinequal** in my notebook.In the next step I import theRandom Forest Classifier and fit it on x-train and y-train set and then predict it on my x-test.With the accuracy or probibility of 78% the test set can predict correct predictions in future .

[21]:
```
rfcWinequal = RandomForestClassifier(n_estimators=200)
rfcWinequal.fit(X_train, y_train)
pred_rfc = rfcWinequal.predict(X_test)
```

[22]:
```
#Let's see how our model performed
print(classification_report(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.72      0.79      0.76       141
           1       0.82      0.76      0.79       179

    accuracy                           0.78       320
   macro avg       0.77      0.78      0.77       320
weighted avg       0.78      0.78      0.78       320
```

[23]:
```
#Confusion matrix for the random forest classification
print(confusion_matrix(y_test, pred_rfc))
```

```
[[112  29]
 [ 43 136]]
```

1.9.2 8.2. Stochastic Gradient Decent Classifier

To explain about Stochastic Gradient Descent classifier(also known SGD) I like to begin with the fact that SGD is a methode for optimizing an object.The next point I would like to add here is that SGD is also counted as a stochastic approximation of gradient descent optimization due to the fact that it is used as an actual gradient .It is worth mentioning that in high-dimensional optimization problems SGd performs very well by achiving faster in this in trade for a lower convergence rate.(https://en.wikipedia.org/wiki/Stochastic_gradient_descent) In the following block I import SGDClassifier and confusion matrix from sklearn library then fit it on my train set and train it with SGD and predict on the test set .The accuracy shows 71% .

[24]:
```
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import confusion_matrix, classification_report
sgd = SGDClassifier(penalty=None)
sgd.fit(X_train, y_train)
pred_sgd = sgd.predict(X_test)
```

[25]:
```
print(classification_report(y_test, pred_sgd))
```

```
              precision    recall  f1-score   support

           0       0.65      0.74      0.69       141
           1       0.77      0.68      0.72       179

    accuracy                           0.71       320
   macro avg       0.71      0.71      0.71       320
weighted avg       0.72      0.71      0.71       320
```

[26]:
```
print(confusion_matrix(y_test, pred_sgd))
```

```
[[105  36]
 [ 57 122]]
```

1.9.3 8.3. SVC()

SVM,A support vector classifier illustrates as a supervised machine learning model.To be more specefic, SVC is a classification algorithms for two-group classification problems.In addition to what I have just said I have to add the point that SVC is able to categorize the next lable after giving this model sets of labled training data for each of the categories. The plus point about SVC when compared to other algorithms like neural network is that SVC not only has higher speed and better performance for working with limited samples but also it is suitable for text classification problems. ref:https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/ In the folloing step I import SVC ,fit it and predict my test set,I got 77% accuracy .

[27]:
```
svc = SVC()
svc.fit(X_train, y_train)
```

```
pred_svc = svc.predict(X_test)
```

[28]:
```
print(classification_report(y_test, pred_svc))
```

```
              precision    recall  f1-score   support

           0       0.72      0.77      0.74       141
           1       0.81      0.76      0.78       179

    accuracy                           0.77       320
   macro avg       0.76      0.77      0.76       320
weighted avg       0.77      0.77      0.77       320
```

1.10 9. Hyperprameter Tuning

I continuemy work with with finding the best parameter for SCV algorithm so I use gridsearch methode,fit it and train my algorithm with it again.

[ ]:
```
#Finding best parameters for our SVC model
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)
```

[32]:
```
grid_svc.fit(X_train, y_train)
```

[32]:
```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': [0.1, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4],
                         'gamma': [0.1, 0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4],
                         'kernel': ['linear', 'rbf']},
             scoring='accuracy')
```

[33]:
```
#Best parameters for our svc model
grid_svc.best_params_
```

[33]:
```
{'C': 1.4, 'gamma': 0.9, 'kernel': 'rbf'}
```

[50]:
```
#Let's run our SVC again with the best parameters.
svc2 = SVC(C = 1.4, 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       0.77      0.67      0.72       141
           1       0.77      0.84      0.80       179

    accuracy                           0.77       320
   macro avg       0.77      0.76      0.76       320
weighted avg       0.77      0.77      0.77       320
```

[41]:
```
print("Confusion Matrix:\n",confusion_matrix(pred_svc2,y_test))
```

```
Confusion Matrix:
 [[ 94  30]
 [ 47 149]]
```

1.11 9. Conclusion

In conclusion, I would like to say that in this notebook I tried to predict the quality of wine from a company with the purpose of predicting the quality for future customers and helping the company has better efficiency in its market. I used steps to prepare my dataset for machine learning algorithms and I tried 3 different models and get the highest accuracy with the SVC model which means that this model can predict the quality of wine with 77 % accuracy in the future.

1.12 10.Conclusion

- https://en.wikipedia.org/wiki/Stochastic_gradient_descent
- :https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/
- https://www.datacamp.com/tutorial/random-foreAdditionallysts-classifier-python)
- https://www.sharpsightlabs.com/blog/seaborn-countplot/
- https://www.educba.com/machine-learning-libraries
- https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009