SVC : Wine DataSet BY Divyani Pathak

Life Cycle of Machine Learning Project

BY Divyani Pathak
LinkedIn profile : https://www.linkedin.com/in/divyani-pathak-5a5a7a1a/
Github Profile : https://github.com/Divyani-Pathak

In this notebook I'm going to do the following:

1. Problem Statement

Using the SVC to predict the quality of the wine

2. Data Collection

a. importing the library

```
# Numpy and Pandas
import numpy as np, pd
# Matplotlib and Seaborn
import matplotlib.pyplot as plt
import seaborn as sns

# Warnings
import warnings
warnings.filterwarnings('ignore')

# Pandas display options
pd.pd.set_option('display.max_columns', None)
pd.pd.set_option('display.max_rows', None)

# Scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, f1_score, precision_score, recall_score, roc_curve, roc_auc_score, RocCurveDisplay
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import VarianceThreshold
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics.cluster import silhouette_score

# Scipy
import scipy.stats as stats
from scipy.stats import chi2_contingency

# Statsmodels
import statsmodels.stats.multicomp as mc
import statsmodels.stats.multitest as smt
import statsmodels.stats.weightstats as sw

# Joblib
import joblib

# Pickle
import pickle

# Warnings
import warnings
warnings.filterwarnings('ignore')
```

b. reading the dataset

```
df = pd.read_csv('https://raw.githubusercontent.com/ybifoundation/Dataset/main/Wine%20Quality%20Dataset.csv')
```

```
df.shape
```

```
(1599, 12)
```

c. check the column

```
df.columns
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype=object)
```

d. check the unique value in quality

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

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

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

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

e. check the full detail

```
df.info()
```

```
<class>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 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   int64
 6   total sulfur dioxide  1599 non-null   int64
 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
```

f. check the datatype

```
df.dtypes
```

```
fixed acidity         float64
volatile acidity      float64
citric acid           float64
residual sugar        float64
chlorides             float64
free sulfur dioxide     int64
total sulfur dioxide    int64
density               float64
pH                    float64
sulphates             float64
alcohol               float64
quality                 int64
dtype: object
```

3. Data Profiling and Data Cleaning

```
df.shape
```

```
(1599, 12)
```

```
df.isnull().sum()
```

```
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
```

a. check the column

```
df.columns
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype=object)
```

b. check the unique value in quality

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

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

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

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

c. check the full detail

```
df.info()
```

```
<class>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 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   int64
 6   total sulfur dioxide  1599 non-null   int64
 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
```

d. check the datatype

```
df.dtypes
```

```
fixed acidity         float64
volatile acidity      float64
citric acid           float64
residual sugar        float64
chlorides             float64
free sulfur dioxide     int64
total sulfur dioxide    int64
density               float64
pH                    float64
sulphates             float64
alcohol               float64
quality                 int64
dtype: object
```

4. Statistical Analysis

4.1 Numerical and categorical features

```
# Numerical features
num = df.select_dtypes(include=['int64','float64'])
print('Numerical features : ', num.shape[1], '\n', list(num.columns))

# Categorical features
cat = df.select_dtypes(include=['object'])
print('Categorical features : ', cat.shape[1], '\n', list(cat.columns))

# Ordinal features : These are categorical features with a meaningful order.
ordinal = ['Your ordinal features']
print('Ordinal features : ', len(ordinal), '\n', ordinal)

# Nominal features : These are categorical features without any order.
nominal = cat.drop(ordinal, axis=1)
print('Nominal features : ', nominal.shape[1], '\n', list(nominal.columns))
```

```
Numerical features :  12
 ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'quality']
Categorical features :  0
 []
Ordinal features :  1
 ['Your ordinal features']
Nominal features :  0
 []
```

4.2 Uni-variant Analysis

In order to do univariate analysis we'll create a function. The purpose is to understand the distribution of values for a single variable.

```
# Numerical features
num = df.select_dtypes(include=['int64','float64'])
print('Numerical features : ', num.shape[1], '\n', list(num.columns))
```

```
Numerical features :  12
 ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'quality']
```

```
df.describe(num.columns).T
```

	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.319637	1.741096	4.60000	7.10000	7.90000	9.200000	15.90000
volatile acidity	1599.0	0.527821	0.179060	0.12000	0.39000	0.52000	0.640000	1.58000
citric acid	1599.0	0.270976	0.194801	0.00000	0.09000	0.26000	0.420000	1.00000
residual sugar	1599.0	2.538806	1.409928	0.90000	1.90000	2.20000	2.600000	15.50000
chlorides	1599.0	0.087467	0.047065	0.01200	0.07000	0.08000	0.090000	0.61100
free sulfur dioxide	1599.0	15.874922	10.460157	1.00000	7.00000	14.00000	21.000000	72.00000
total sulfur dioxide	1599.0	46.467792	32.895324	6.00000	22.00000	38.00000	62.000000	289.00000
density	1599.0	0.996747	0.001887	0.99007	0.99356	0.99675	0.997845	1.00369
pH	1599.0	3.311113	0.154386	2.74000	3.21000	3.31000	3.400000	4.01000
sulphates	1599.0	0.658149	0.169507	0.33000	0.55000	0.62000	0.730000	2.00000
alcohol	1599.0	10.422983	1.065668	8.40000	9.50000	10.20000	11.100000	14.90000
quality	1599.0	5.636023	0.807569	3.00000	5.00000	6.00000	6.000000	8.00000

```
df.describe(num.columns).T.to_csv('Numerical Features Analysis.csv')
```

Numerical Features Analysis (uni-variant analysis) - KDE Plot and Scatter and etc.

```
# Numerical features
num = df.select_dtypes(include=['int64','float64'])
print('Numerical features : ', num.shape[1], '\n', list(num.columns))
```

```
Numerical features :  12
 ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'quality']
```

```
# Numerical features : Uni-variant analysis
num = df.select_dtypes(include=['int64','float64'])
print('Numerical features : ', num.shape[1], '\n', list(num.columns))
for i in num.columns:
    plt.figure(figsize=(15,5))
    plt.suptitle('Uni-variant analysis of Numerical Features', fontsize=20, fontweight='bold', alpha=0.8, y=1.)
    ax = plt.subplot(1,3,1)
    sns.boxplot(num[i], palette='Blues')
    ax = plt.subplot(1,3,2)
    sns.kde
```

Scatter Plot

```
# Scatter plot
# Scatter plot with pairs Numerical Features to analyze feature
num = df.select_dtypes(include=['int64','float64'])
print('Numerical features : ', num.shape[1], '\n', list(num.columns))
sns.pairplot(data=num,
             palette='Blues',
             corner=True,
             diag_kind='kde',
             kind='scatter',
             plot_kws={'s':80,
                       'alpha':0.7,
                       'linewidth':0.5,
                       'edgecolor':'black'},
             diag_kws={'shade':True,
                       'linewidth':3,
                       'alpha':0.7,
                       'bw_adjust':5,
                       'cut':0,
                       'multiple':'stack',
                       'palette':'Blues'})
plt.suptitle('Scatter plot with pairs Numerical Features to analyze feature', fontsize=20, fontweight='bold', alpha=0.8, y=1.)
```

count plot for target feature

```
# count plot for target feature
num = df.select_dtypes(include=['int64','float64'])
print('Numerical features : ', num.shape[1], '\n', list(num.columns))
cat = df.select_dtypes(include=['object'])
print('Categorical features : ', cat.shape[1], '\n', list(cat.columns))
for i in cat.columns:
    plt.figure(figsize=(15,5))
    plt.suptitle('Uni-variant analysis of categorical binary/ multi-var Features', fontsize=20, fontweight='bold', alpha=0.8, y=1.)
    ax = plt.subplot(1,3,1)
    sns.countplot(cat[i], palette='Blues')
    ax = plt.subplot(1,3,2)
    sns.countplot(cat[i], palette='Blues', hue=df['quality'])
    ax = plt.subplot(1,3,3)
    df.groupby(cat[i])['quality'].mean().sort_values(ascending=False).plot.bar(color=sns.color_palette('Blues'))
    plt.xticks(rotation=90)
```

Box Plot

```
# Box Plot
num = df.select_dtypes(include=['int64','float64'])
print('Numerical features : ', num.shape[1], '\n', list(num.columns))
cat = df.select_dtypes(include=['object'])
print('Categorical features : ', cat.shape[1], '\n', list(cat.columns))
for i in num.columns:
    plt.figure(figsize=(15,5))
    plt.suptitle('Uni-variant analysis of numerical features', fontsize=20, fontweight='bold', alpha=0.8, y=1.)
    ax = plt.subplot(1,3,1)
    sns.boxplot(num[i], palette='Blues')
    ax = plt.subplot(1,3,2)
    sns.boxplot(num[i], palette='Blues', hue=df['quality'])
    ax = plt.subplot(1,3,3)
    df.groupby(df['quality'])[i].mean().sort_values(ascending=False).plot.bar(color=sns.color_palette('Blues'))
    plt.xticks(rotation=90)
```


4.3 Bi-variant Analysis

Key Plot for bi-variate analysis numerical vs categorical features

```
# Bi-variant Analysis
num = df.select_dtypes(include=['int64','float64'])
print('Numerical features : ', num.shape[1], '\n', list(num.columns))
cat = df.select_dtypes(include=['object'])
print('Categorical features : ', cat.shape[1], '\n', list(cat.columns))
for i in num.columns:
    plt.figure(figsize=(15,5))
    plt.suptitle('Strip Plot', fontsize=20, fontweight='bold', alpha=0.8, y=1.)
    ax = plt.subplot(1,3,1)
    sns.stripplot(df[cat[cat.columns[0]].unique()[0]][i], palette='Blues')
    ax = plt.subplot(1,3,2)
    sns.stripplot(df[cat[cat.columns[0]].unique()[0]][i], palette='Blues', hue=df['quality'])
    ax = plt.subplot(1,3,3)
    sns.stripplot(df[cat[cat.columns[0]].unique()[1]][i], palette='Blues')
    ax = plt.subplot(1,3,4)
    sns.stripplot(df[cat[cat.columns[0]].unique()[1]][i], palette='Blues', hue=df['quality'])
```


4.4 Multivariant Analysis

```
df.corr()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.390558
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.226373
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732
chlorides	0.093705	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.050656
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000

pair plot

```
# pair plot
num = df.select_dtypes(include=['int64','float64'])
sns.pairplot(data=num,
             palette='Blues',
             corner=True,
             diag_kind='kde',
             kind='scatter',
             plot_kws={'s':80,
                       'alpha':0.7,
                       'linewidth':0.5,
                       'edgecolor':'black'},
             diag_kws={'shade':True,
                       'linewidth':3,
                       'alpha':0.7,
                       'bw_adjust':5,
                       'cut':0,
                       'multiple':'stack',
                       'palette':'Blues'})
```


Heat map

```
# Heat map
num = df.select_dtypes(include=['int64','float64'])
plt.figure(figsize=(15,5))
sns.heatmap(num.corr(),
            annot=True,
            cmap='Blues',
            linewidths=0.5,
            linecolor='black',
            annot_kws={'size':10,
                       'weight':'bold',
                       'rotation':0})
```

Model Building

```
df.shape
```

```
(1599, 12)
```

```
df.dtypes
```

```
fixed acidity         float64
volatile acidity      float64
citric acid           float64
residual sugar        float64
chlorides             float64
free sulfur dioxide     int64
total sulfur dioxide    int64
density               float64
pH                    float64
sulphates             float64
alcohol               float64
quality                 int64
dtype: object
```

1. Independent and Dependent Features

```
# X and y data
X = df.drop('quality',axis=1)
y = df['quality']
```

```
# X and y data shape
X.shape, y.shape
```

```
((1599, 11), (1599,))
```

2. Splitting the data into Train and Test data set

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0, stratify=y)
```

```
X_train.shape
```

```
(1119, 11)
```

```
X_test.shape
```

```
(480, 11)
```

```
y_train.shape
```

```
(1119,)
```

```
y_test.shape
```

```
(480,)
```

3. Standardize or feature scaling the datasets

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
X_train[numerical_features] = sc.fit_transform(X_train[numerical_features])
X_test[numerical_features] = sc.transform(X_test[numerical_features])
```

```
X_train[numerical_features].sample(5).T # print data leakage and scaling
```

	1000	1337	1132	1235	1004
fixed acidity	0.0000000	-1.0000000	-0.5000000	-0.2500000	0.0000000
volatile acidity	0.5000000	-0.5000000	0.2500000	0.2500000	0.5000000
citric acid	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000
residual sugar	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000
chlorides	0.5000000	-0.5000000	0.5000000	0.5000000	0.5000000
free sulfur dioxide	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000
total sulfur dioxide	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000
density	0.5000000	-0.5000000	0.5000000	0.5000000	0.5000000
pH	-0.5000000	0.5000000	-0.5000000	-0.5000000	-0.5000000
sulphates	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000
alcohol	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000

```
X_test[numerical_features].sample(5).T
```

	1000	1337	1132	1235	1004
fixed acidity	0.0000000	-1.0000000	-0.5000000	-0.2500000	0.0000000
volatile acidity	0.5000000	-0.5000000	0.2500000	0.2500000	0.5000000
citric acid	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000
residual sugar	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000
chlorides	0.5000000	-0.5000000	0.5000000	0.5000000	0.5000000
free sulfur dioxide	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000
total sulfur dioxide	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000
density	0.5000000	-0.5000000	0.5000000	0.5000000	0.5000000
pH	-0.5000000	0.5000000	-0.5000000	-0.5000000	-0.5000000
sulphates	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000
alcohol	-0.5000000	-0.5000000	-0.5000000	-0.5000000	-0.5000000

```
[0.0]]
[[ 0.0000000e+00, 0.0000000e+00, 0.0000000e+00, ...,
  0.0000000e+00, 0.0000000e+00, 0.0000000e+00],
 [
```