## Support Vector Machine for glass dataset

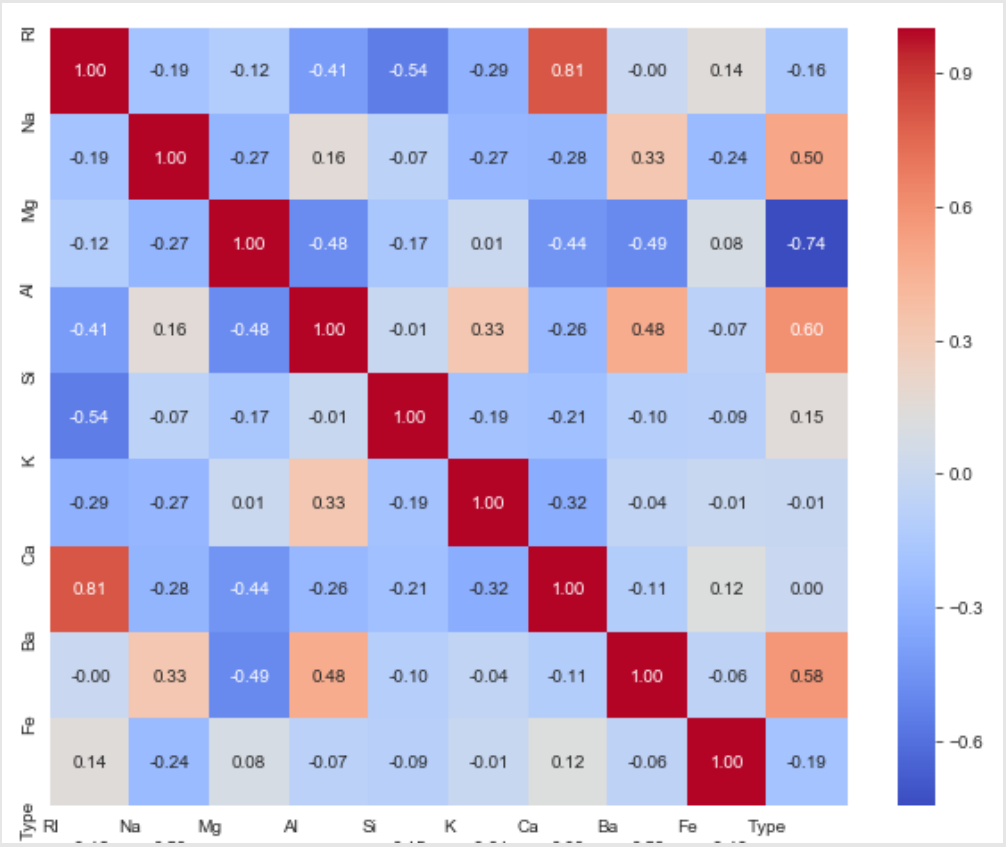
- In machine learning, support-vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
- SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.
- An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.
- New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

**Code:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from collections import Counter
from IPython.core.display import display, HTML
sns.set_style('darkgrid')

#reading dataset
dataset = pd.read_csv('D:/academics/MLpractice/glass.csv')
dataset.head()
```

| | RI | Na | Mg | Al | Si | K | Ca | Ba | Fe | Type |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.52101 | 13.64 | 4.49 | 1.10 | 71.78 | 0.06 | 8.75 | 0.0 | 0.0 | 1 |
| 1 | 1.51761 | 13.89 | 3.60 | 1.36 | 72.73 | 0.48 | 7.83 | 0.0 | 0.0 | 1 |
| 2 | 1.51618 | 13.53 | 3.55 | 1.54 | 72.99 | 0.39 | 7.78 | 0.0 | 0.0 | 1 |
| 3 | 1.51766 | 13.21 | 3.69 | 1.29 | 72.61 | 0.57 | 8.22 | 0.0 | 0.0 | 1 |
| 4 | 1.51742 | 13.27 | 3.62 | 1.24 | 73.08 | 0.55 | 8.07 | 0.0 | 0.0 | 1 |

```
#plot dataset
corr = dataset.corr()
#Plot figsize
fig, ax = plt.subplots(figsize=(10, 8))
#Generate Heat Map, allow annotations and place floats in map
sns.heatmap(corr, cmap='coolwarm', annot=True,
fmt=".2f")
#Apply xticks
plt.xticks(range(len(corr.columns)), corr.columns);
#Apply yticks
plt.yticks(range(len(corr.columns)), corr.columns)
#show plot
plt.show()
```



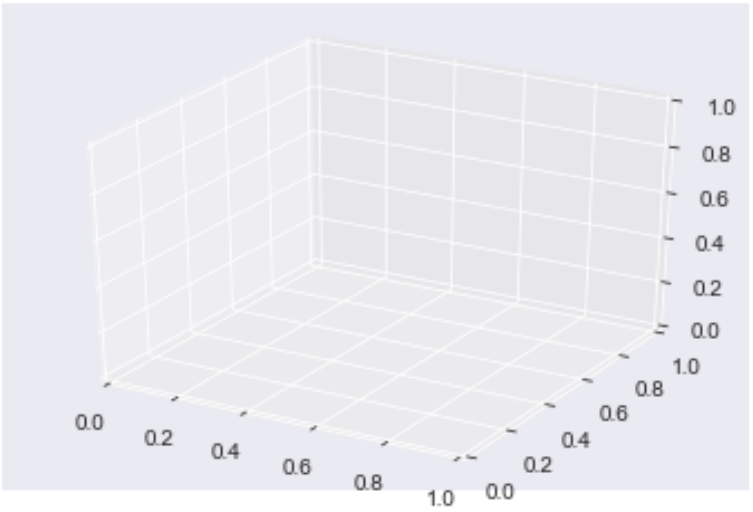

```
# Splitting the dataset into the Training set and Test
X = dataset.drop('Type', axis = 1).values
y = dataset['Type'].values.reshape(-1,1)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size = 0.25, random_state = 42)

#SVM
from sklearn.svm import SVC
clf_svm = SVC()
clf_svm.fit(X_train, y_train)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

```
#Classification Report, Confusion Matrix and accuracy
y_pred = clf_svm.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test, y_pred))
print('\n')
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.50      1.00      0.67        14
           2       0.77      0.48      0.59        21
           3       0.00      0.00      0.00         4
           5       0.75      0.75      0.75         4
           6       0.50      0.33      0.40         3
           7       1.00      0.88      0.93         8

    accuracy                           0.65        54
   macro avg       0.59      0.57      0.56        54
weighted avg       0.66      0.65      0.62        54



[[14  0  0  0  0  0]
 [10 10  0  1  0  0]
 [ 4  0  0  0  0  0]
 [ 0  1  0  3  0  0]
 [ 0  2  0  0  1  0]
 [ 0  0  0  0  1  7]]
```

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
types = dataset.Type.unique()
train = pd.DataFrame()
test = pd.DataFrame()
fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')
for i in range(len(types)+1):
 count = i+1
 train_tempt = train.loc[train['Type'] == count]
 x = train_tempt['Mg']
 y = train_tempt['Al']
 z = train_tempt['K']

 ax.scatter(x, y, z, c= [float(i)/float(len(types)),
0.0, float(len(types)-i)/float(len(types))],
marker='o')

 ax.set_xlabel(str('Mg'))
 ax.set_ylabel(str('Al'))
 ax.set_zlabel(str('K'))
plt.show()
```




#AUC Curve

```
from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred,
pos_label=2)
metrics.auc(fpr, tpr)
```

```
0.4437229437229437
```

```
plt.plot(fpr,tpr)
plt.show()
```