Classify the Size Categorie using SVM

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import silhouette_score, calinski_harabasz_score, silhouette_samples
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
```

```
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/forestfires.csv')
```

```
data
```

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	monthjul	monthjun	monthmar	monthmay	monthnov
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0	0	0	0	1	0	0
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	...	0	0	0	0	0	0	0
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0	0	0	0	0	0	0
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	...	0	0	0	0	1	0	0
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	...	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
512	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	...	0	0	0	0	0	0	0
513	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	...	0	0	0	0	0	0	0
514	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	...	0	0	0	0	0	0	0
515	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	...	0	0	0	0	0	0	0
516	nov	tue	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	...	0	0	0	0	0	0	1

517 rows × 31 columns

```
data.info()
```

```
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 517 entries, 0 to 516
    Data columns (total 31 columns):
     #   Column         Non-Null Count  Dtype
    ---  ------         --------------  -----
     0   month          517 non-null    object
     1   day            517 non-null    object
     2   FFMC           517 non-null    float64
     3   DMC            517 non-null    float64
     4   DC             517 non-null    float64
     5   ISI            517 non-null    float64
     6   temp           517 non-null    float64
     7   RH             517 non-null    int64
     8   wind           517 non-null    float64
     9   rain           517 non-null    float64
     10  area           517 non-null    float64
     11  dayfri         517 non-null    int64
     12  daymon         517 non-null    int64
     13  daysat         517 non-null    int64
     14  daysun         517 non-null    int64
     15  daythu         517 non-null    int64
     16  daytue         517 non-null    int64
     17  daywed         517 non-null    int64
```

```
 18  monthapr       517 non-null    int64
 19  monthaug       517 non-null    int64
 20  monthdec       517 non-null    int64
 21  monthfeb       517 non-null    int64
 22  monthjan       517 non-null    int64
 23  monthjul       517 non-null    int64
 24  monthjun       517 non-null    int64
 25  monthmar       517 non-null    int64
 26  monthmay       517 non-null    int64
 27  monthnov       517 non-null    int64
 28  monthoct       517 non-null    int64
 29  monthsep       517 non-null    int64
 30  size_category  517 non-null    object
dtypes: float64(8), int64(20), object(3)
memory usage: 125.3+ KB
```

```
data.head()
```

	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	...	monthfeb	monthjan	monthjul	monthjun	monthmar	monthmay	monthnov	mon
0	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	...	0	0	0	0	1	0	0	
1	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	...	0	0	0	0	0	0	0	
2	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	...	0	0	0	0	0	0	0	
3	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	...	0	0	0	0	1	0	0	
4	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	...	0	0	0	0	1	0	0	

5 rows × 31 columns

```
data.isnull().sum()
```

```
month            0
day              0
FFMC             0
DMC              0
DC               0
ISI              0
temp             0
RH               0
wind             0
rain             0
area             0
dayfri           0
daymon           0
daysat           0
daysun           0
daythu           0
daytue           0
daywed           0
monthapr         0
monthaug         0
monthdec         0
monthfeb         0
monthjan         0
monthjul         0
monthjun         0
monthmar         0
monthmay         0
monthnov         0
monthoct         0
monthsep         0
size_category    0
dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
data['size_category'] = label_encoder.fit_transform(data['size_category'])
```

```
X =  data.drop("size_category", axis=1)
y =  data["size_category"]

# Convert categorical variables ---(month and day)---- into numerical
X = pd.get_dummies(X, columns=["month", "day"])
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 0)
```

```
cols = X_train.columns
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train = pd.DataFrame(X_train, columns=[cols])
X_test = pd.DataFrame(X_test, columns=[cols])
X_train.describe()
```

	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	da
count	4.130000e+02	4.130000e+02	4.130000e+02	4.130000e+02	4.130000e+02	4.130000e+02	4.130000e+02	4.130000e+02	4.130000e+02	4.1
mean	9.419422e-16	1.333343e-16	8.602212e-17	-6.021549e-17	4.537667e-16	1.849476e-16	-3.010774e-17	-3.440885e-17	3.440885e-17	3.
std	1.001213e+00	1.001213e+00	1.001213e+00	1.001213e+00	1.001213e+00	1.001213e+00	1.001213e+00	1.001213e+00	1.001213e+00	1.0
min	-1.260458e+01	-1.704887e+00	-2.138457e+00	-1.939306e+00	-2.876600e+00	-1.763658e+00	-1.709652e+00	-7.126589e-02	-2.320156e-01	-
25%	-6.158922e-02	-7.802221e-01	-4.426625e-01	-5.482765e-01	-5.651456e-01	-7.333164e-01	-7.153197e-01	-7.126589e-02	-2.320156e-01	-
50%	1.661467e-01	-5.127821e-02	4.766767e-01	-1.416679e-01	9.034143e-02	-1.878417e-01	2.808847e-03	-7.126589e-02	-2.266325e-01	-
75%	3.938826e-01	4.726984e-01	6.711908e-01	4.147439e-01	6.768298e-01	5.394579e-01	4.999748e-01	-7.126589e-02	-8.756855e-02	-
max	9.719814e-01	2.767408e+00	1.274345e+00	1.006635e+01	2.453545e+00	3.388048e+00	2.985804e+00	1.955061e+01	1.650679e+01	2.3

8 rows × 47 columns

```
# import SVC classifier

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score
svc=SVC()
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Model accuracy score with default hyperparameters: {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with default hyperparameters: 0.7596
```

```
# instantiate classifier with rbf kernel and C=100
svc=SVC(C=100.0)
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Model accuracy score with rbf kernel and C=100.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with rbf kernel and C=100.0 : 0.8750
```

```
# instantiate classifier with rbf kernel and C=1000
svc=SVC(C=1000.0)
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
print('Model accuracy score with rbf kernel and C=1000.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with rbf kernel and C=1000.0 : 0.8654
```

```
# instantiate classifier with linear kernel and C=1.0
linear_svc=SVC(kernel='linear', C=1.0)
# fit classifier to training set
linear_svc.fit(X_train,y_train)
# make predictions on test set
y_pred_test=linear_svc.predict(X_test)
# compute and print accuracy score
print('Model accuracy score with linear kernel and C=1.0 : {0:0.4f}'.
format(accuracy_score(y_test, y_pred_test)))
```

```
     Model accuracy score with linear kernel and C=1.0 : 0.9231
```

```
# instantiate classifier with linear kernel and C=100.0
linear_svc100=SVC(kernel='linear', C=100.0)
linear_svc100.fit(X_train, y_train)
y_pred=linear_svc100.predict(X_test)
print('Model accuracy score with linear kernel and C=100.0 : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
     Model accuracy score with linear kernel and C=100.0 : 0.9712
```

```
# instantiate classifier with linear kernel and C=1000.0
linear_svc1000=SVC(kernel='linear', C=1000.0)
linear_svc1000.fit(X_train, y_train)
y_pred=linear_svc1000.predict(X_test)
print('Model accuracy score with linear kernel and C=1000.0 : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
     Model accuracy score with linear kernel and C=1000.0 : 0.9327
```

```
##Compare the train-set and test-set accuracy
y_pred_train = linear_svc.predict(X_train)
y_pred_train
```

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

```
print('Training-set accuracy score: {0:0.4f}'.
format(accuracy_score(y_train, y_pred_train)))
```

```
     Training-set accuracy score: 0.9516
```

```
##   Check for overfitting and underfitting

print('Training set score: {:.4f}'.format(linear_svc.score(X_train,
y_train)))
print('Test set score: {:.4f}'.format(linear_svc.score(X_test,
y_test)))
```

```
     Training set score: 0.9516
     Test set score: 0.9231
```