Supervised Machine Learning Methods

Ildem Sanli
Vito Coquet

Data Processing

- We started by dropping the columns that we thought were not relevant for our models
- Then we converted the ‘rating’, ‘vote’ and ‘runtime’ columns to numeric
- We made the ‘kind’ column more uniform by grouping movies and series together
- We decided to keep only the first country of each cell in the corresponding column, same for the ‘genre’ column
- We kept only the 10 most occurring values in the ‘country’ column
- We binned the rating column into 5 equal parts
- We encoded the categorical columns
- And finally we scaled the vote and runtime columns

Number of movies per year

```python
moviesPerYear = net_c['year'].value_counts()
moviesPerYear
```

✓ 0.1s

```
2003.0    445
2002.0    423
2004.0    405
2001.0    398
2000.0    348
         ...
1916.0      1
1919.0      1
1933.0      1
1922.0      1
1923.0      1
Name: year, Length: 84, dtype: int64
```

Movies per country

```
moviesPerCountry = net['country'].value_counts()
moviesPerCountry
```

✓ 0.1s

```
'United States'     4180
'United Kingdom'    1058
Other                776
'Japan'              607
'Canada'             373
'France'             349
Unknown              294
'Hong Kong'          265
'India'              254
'Italy'              152
'Germany'            110
Name: country, dtype: int64
```

The most popular genre per year

```python
genrePerYear = net.groupby(['genre'])['year'].value_counts().unstack().idxmax()
genrePerYear
```

✓ 0.3s

```
year
1905.0    'Documentary'
1910.0    'Documentary'
1913.0            Other
1914.0         'Action'
1916.0          'Drama'
              ...
2001.0    'Documentary'
2002.0    'Documentary'
2003.0    'Documentary'
2004.0    'Documentary'
2005.0    'Documentary'
Length: 91, dtype: object
```

The most popular genre per country

```
genrePerCountry = net.groupby(['genre'])['country'].value_counts().unstack().idxmax()
genrePerCountry
```

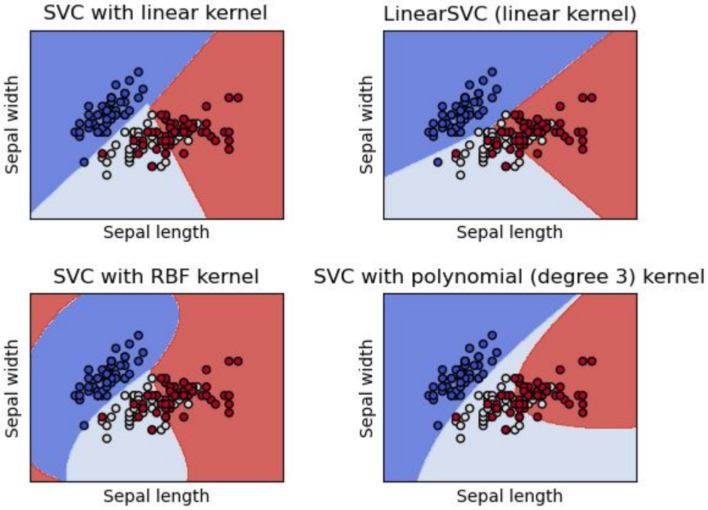
✓ 0.5s

```
country
'Canada'                    'Drama'
'France'                    'Drama'
'Germany'                   'Drama'
'Hong Kong'                'Action'
'India'                     'Drama'
'Italy'                     'Drama'
'Japan'                 'Animation'
'United Kingdom'            'Drama'
'United States'       'Documentary'
Other                       'Drama'
Unknown               'Documentary'
dtype: object
```

SVC (C-support vector classification)

SVC is a class of Support Vector Machines (SVM).

SVMs divide the datasets into number of classes by generating a hyperplane.



- generate hyperplanes iteratively that separates the classes in the best way

- choose the hyperplane that segregate the classes correctly

SVC (C-support vector classification)

```
SVC metrics
-----------

Accuracy: 0.64
ROC-AUC score: 0.47
Confusion matrix
 [[  0  54   0   0   0]
 [  0 705   0   0   0]
 [  0 261   0   0   0]
 [  0   5   0   0   0]
 [  0  84   0   0   0]]
Precision score: 0.4
Recall score: 0.66
F1 score: 0.49
```

Categorical Naïve Bayes

Probabilistic classifiers based on Bayes Theorem

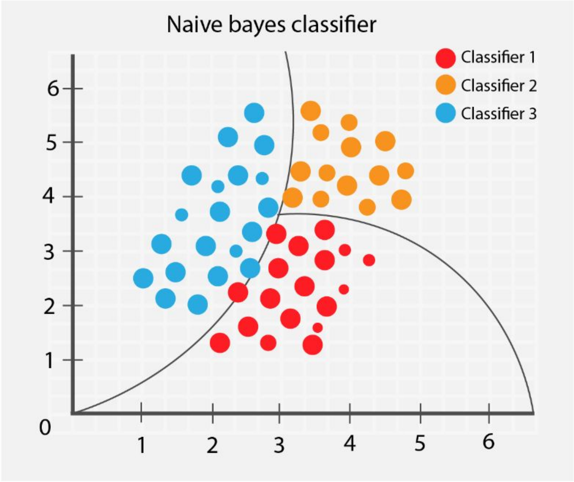
Strong independence assumption between the features

Suitable for classification with discrete features that are categorically distributed.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$



Categorical Naïve Bayes

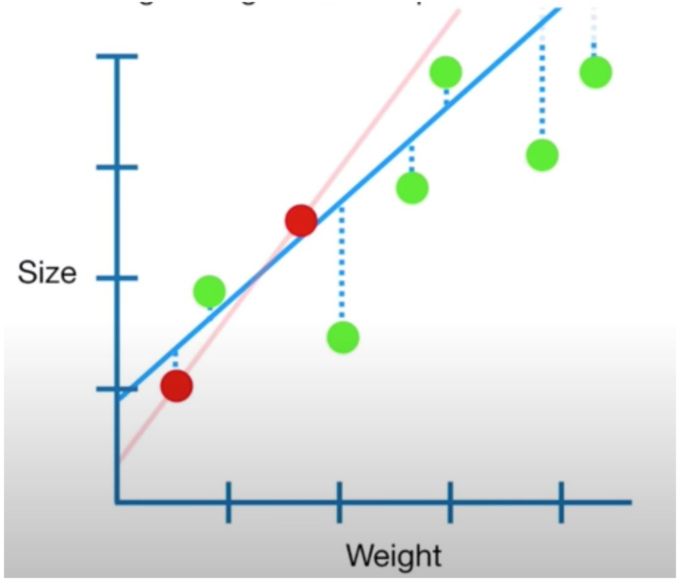
```
CategoricalNB
----------

Accuracy: 0.66
ROC-AUC score: 0.75
Confusion matrix
 [[  0  24  30   0   0]
 [  0 662  27   0  16]
 [  0 210  50   0   1]
 [  0   3   2   0   0]
 [  0  66   3   0  15]]
Precision score: 0.58
Recall score: 0.66
F1 score: 0.59
```

Ridge Classifier

Converts the target values into {-1, 1} and then treats the problem as a regression task (multi-output regression in the multiclass case).

Adds bias to a multilinear regression model to get more accurate regression with tested data at a cost of losing accuracy for the training data.



Ridge Classifier

```
RidgeClassifier
----------

Accuracy: 0.64
Confusion matrix
 [[  0  54   0   0   0]
 [  0 705   0   0   0]
 [  0 261   0   0   0]
 [  0   5   0   0   0]
 [  0  83   0   0   1]]
Precision score: 0.48
Recall score: 0.64
F1 score: 0.5
```

RFE for RidgeClassifier

['kind', 'vote', 'language', 'runtime']

```
RidgeClassifier
----------

Accuracy: 0.64
Confusion matrix
 [[  0  54   0   0   0]
 [  0 705   0   0   0]
 [  0 261   0   0   0]
 [  0   5   0   0   0]
 [  0  83   0   0   1]]
Precision score: 0.48
Recall score: 0.64
F1 score: 0.5
```

Extra Trees Classifier

Large number of decision trees where the final decision is obtained taking into account the prediction of every tree.

Random split - reduces the variance of the model a bit more, at the expense of a slightly greater increase in bias.

```python
#### ExtraTreesClassifier

etc=ExtraTreesClassifier(n_estimators=1000, max_depth=10).fit(x_train, y_train)
y_pred=etc.predict(x_test)
acc=etc.score(x_test, y_test)
```

```
ExtraTreesClassifier

----------


Accuracy: 0.65

ROC-AUC score: 0.78

Confusion matrix
 [[  0  46   8   0   0]
 [  0 694   9   0   2]
 [  0 246  15   0   0]
 [  0   3   2   0   0]
 [  0  77   0   0   7]]
Precision score: 0.58

Recall score: 0.65

F1 score: 0.53
```

```
#### ExtraTreesClassifier

etc=ExtraTreesClassifier(n_estimators=3000, max_depth=50).fit(x_train, y_train)
y_pred=etc.predict(x_test)
acc=etc.score(x_test, y_test)
```

```
ExtraTreesClassifier
----------


Accuracy: 0.68
ROC-AUC score: 0.84
Confusion matrix
 [[ 12  20  22   0   0]
 [  3 622  60   0  20]
 [  6 163  87   0   5]
 [  2   1   2   0   0]
 [  0  54   1   0  29]]
Precision score: 0.64
Recall score: 0.68
F1 score: 0.65
```

RFE for ExtraTreesClassifier ['year', 'vote', 'director', 'runtime']

```
(n_estimators=1000, max_depth=10)
```

```
ExtraTreesClassifier
----------


Accuracy: 0.64
ROC-AUC score: 0.68
Confusion matrix
 [[  0  54   0   0   0]
 [  1 703   0   0   1]
 [  0 261   0   0   0]
 [  0   5   0   0   0]
 [  0  82   0   0   2]]
Precision score: 0.45
Recall score: 0.64
F1 score: 0.5
```

```
(n_estimators=3000, max_depth=50)
```

```
ExtraTreesClassifier
----------


Accuracy: 0.61
ROC-AUC score: 0.62
Confusion matrix
 [[  0  43  10   1   0]
 [ 12 619  62   0  12]
 [  5 200  50   0   6]
 [  0   4   1   0   0]
 [  1  65   9   0   9]]
Precision score: 0.54
Recall score: 0.61
F1 score: 0.55
```