Support Vector Machine

By: Savion Ponce, Juliann Groglio, Tavianne Kemp, Paul Polsinelli

Overview

- Utilizing the SVM algorithm with the Cars93 dataset
- SVM is a supervised learning model with associated learning algorithms that analyze data for classification and regression analysis
- Predicting whether a car has a manual or an automatic transmission
- Based on numeric columns: Price, MPG.city, MPG.highway, Cylinders, EngineSize, Horsepower RPM, Fuel.tank.capacity, and Weight
- Categorical column: Man.trans.avail

Code Samples

[1]:
```
from msilib.schema import Class
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegressionCV
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn import preprocessing
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
import time
startTime = time.time()

df = pd.read_csv('Cars93.csv')
df.head()
```

[3]:
```
df.drop(columns=['Unnamed: 0','Manufacturer','Model','Type','Min.Price','Max.Price','AirBags','DriveTrain','Rev.per.mile','Passengers','Length','Wheelbase','Width'
df.head()
```

[5]:
```
df.drop(df[df['Cylinders'] == 'rotary'].index, inplace = True)
```

[6]:
```
numericData = df.drop(columns='Man.trans.avail').copy()
# Man.trans.avail as category column
categoricalData = df['Man.trans.avail'].copy()

categoricalData = categoricalData.map({'Yes': 1, 'No': 0})
# Make it the column is predicts
```

[7]:
```
df = pd.concat([numericData, categoricalData], axis=1)
```

Code Samples

[8]:
```
x = numericData
x_scaled = preprocessing.scale(x)
x = x_scaled
y = categoricalData
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.25, random_state=0)
```

[9]:
```
Classifier = SVC(kernel='rbf', C=2, gamma='scale')
Classifier.fit(x_train, y_train)
y_pred = Classifier.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)*100
confusion_mat = confusion_matrix(y_test, y_pred)
score = Classifier.score(x_train, y_train)
cv_scores = cross_val_score(Classifier, x_train, y_train, cv=10)
cr = classification_report(y_test, y_pred)
```

10]:
```
print(df.corr())
print('------------------------------------------------------------')
print("Accuracy for SVM is: ", accuracy)
print('------------------------------------------------------------')
print("Confusion Matrix")
print(confusion_mat)
print('------------------------------------------------------------')
print("Score: ", score)
print('------------------------------------------------------------')
print("CV average score: %.2f" % cv_scores.mean())
print('------------------------------------------------------------')
print('Classification Report:')
print(cr)

executionTime = (time.time() - startTime)
print('Execution time in seconds: ' + str(executionTime))
```

Process

1. Imported necessary libraries
2. Read the Cars93 file using read_csv('Cars93')
3. Created a copy of our DataFrame (df)
4. Cleaning Data (Removed irrelevant columns/rows)
5. Created variables 'numericData' and 'categoricalData'
    a. Set numericData equal to all columns except 'Mans.trans.avail'
    b. Set categoricalData equal to 'Mans.trans.avail'
    c. Mapped categoricalData to make 'Yes' = 1 and 'No' = 0
6. Used *concat* to combine the two to restore DataFrame
7. Set x = numericalData, and Set y = categoricalData
8. Conducted our x, y trains and tests using a test size of 0.25

Process (continued)

8. Implemented SVC algorithm with default parameters
    a. kernel = ‘rbf’
    b. C=1
    c. gamma = ‘scaled’
    d. Used fit() operator on x_train and y_train
9. Conducted the predictions using x_test
10. Conducted calculation for accuracy by taking in (y_test, y_pred) * 100
11. Conducted calculation for confusion matrix using (y_test, y_pred)
12. Printed the correlation matrix, confusion matrix, accuracy, score, CV (cross-validation) average, and classification report

Model Evaluation

	Price	MPG.city	MPG.highway	EngineSize	Horsepower	\
Price	1.000000	-0.589267	-0.556696	0.629168	0.784143	
MPG.city	-0.589267	1.000000	0.943668	-0.734536	-0.670457	
MPG.highway	-0.556696	0.943668	1.000000	-0.646166	-0.618626	
EngineSize	0.629168	-0.734536	-0.646166	1.000000	0.790067	
Horsepower	0.784143	-0.670457	-0.618626	0.790067	1.000000	
RPM	-0.036347	0.395603	0.339630	-0.535751	-0.011519	
Fuel.tank.capacity	0.614047	-0.811147	-0.784466	0.786021	0.709805	
Weight	0.658537	-0.850998	-0.816242	0.849277	0.765391	
Man.trans.avail	-0.341525	0.455347	0.412995	-0.632104	-0.380429	

	RPM	Fuel.tank.capacity	Weight	Man.trans.avail
Price	-0.036347	0.614047	0.658537	-0.341525
MPG.city	0.395603	-0.811147	-0.850998	0.455347
MPG.highway	0.339630	-0.784466	-0.816242	0.412995
EngineSize	-0.535751	0.786021	0.849277	-0.632104
Horsepower	-0.011519	0.709805	0.765391	-0.380429
RPM	1.000000	-0.366735	-0.431383	0.425069
Fuel.tank.capacity	-0.366735	1.000000	0.902984	-0.495505
Weight	-0.431383	0.902984	1.000000	-0.611774
Man.trans.avail	0.425069	-0.495505	-0.611774	1.000000

```
------------------------------------------------------------
Accuracy for SVM is:  86.95652173913044
------------------------------------------------------------
Confusion Matrix
[[ 6  1]
 [ 2 14]]
------------------------------------------------------------
Score:  0.8695652173913043
------------------------------------------------------------
CV average score: 0.81
------------------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.86      0.80         7
           1       0.93      0.88      0.90        16

    accuracy                           0.87        23
   macro avg       0.84      0.87      0.85        23
weighted avg       0.88      0.87      0.87        23

Execution time in seconds: 0.24762678146362305
```

Model Comparison

SVM

Accuracy:
86.95652173913044

Time:
0.24762678146362305

Gaussian NB

Accuracy:
82.6086956521739

Time:
0.15183520317077637

- Gaussian NB accuracy lower possibly due to assumption of independence not being upheld.
- Time was lower for Gaussian NB, this could be because the code was more simple.

Improving Performance

- We scaled x using preprocessing attribute of the sklearn module.
- Kept default kernel = rbf, after trying linear
- Changed C from C=1 to C=2
- Kept default gamma to ‘scale’, after trying ‘auto’

Sources

Kim, S. (2022, July 21). GitHub - seungmin478/Machine-Learning. GitHub. Retrieved July 22, 2022, from https://github.com/seungmin478/Machine-Learning

Raparla, A. (2018, September 14). Cars93. Kaggle. Retrieved February 22, 2022, from https://www.kaggle.com/datasets/anand0427/cars93

Wikipedia contributors. (2022, June 20). Support-vector machine. Wikipedia. Retrieved July 22, 2022, from https://en.wikipedia.org/wiki/Support-vector_machine