Support Vector Machines – Features

Emanuel Zaymus

In this report I would like to compare 3 different scenarios for feature configuration. I will cope with encoding categorical features and test whether including PREVIOUS PREDICTED CLASS FEATURE adds value and or not.

Dataset

Kyoto – Normal ADL Activities

Used library

`sklearn.svm.`**`SVC`** with defaults: `kernel='rbf', C=1.0, gamma='scale'`

Scaling

`sklearn.preprocessing.`**`StandardScaler`**

`sklearn.preprocessing.`**`RobustScaler`**

In the last report I showed that the normalization does not have any impact on the final accuracy score of the classifier. On the other hand, the best results were achieved after scaling the data, namely with the `StandardScaler` and `RobustScaler`. That is way I decided to use them (with default settings).

Testing

`sklearn.model_selection.`**`KFold`**

Because of the comparison-to-be I need to test the results **with and without shuffling the feature vectors.** We will see that it is a great difference if we test with or without shuffling.

With shuffling: **`KFold`**`(n_splits=5, shuffle=True, random_state=0)`

Without shuffling: **`KFold`**`(n_splits=5, shuffle=False)`

SCENARIO 1 – Considering all features to be continuous

Features

1. SECONDS FROM MIDNIGHT – of the first record in the window
2. ***DAY OF THE WEEK – MON..SUN => 1..7 - of the last record in the window***
3. SECONDS ELAPSED – between the last and the first record of the window

4.-28. SIMPLE COUNTS OF THE SENSORS

As we can see I leave one categorical feature – DAY OF THE WEEK – in the feature vectors. For comparative reason I am going to treat it as a continuous.

In the *Table 1* we can see that the shuffling the data before testing has quite a big effect on the accuracy. **The accuracy is high despite the wrong handling of the categorical feature.**

Table 1 – Scenario 1 – Accuracy scores (%)

Window size	Without shuffling			With shuffling		
	No scaling	Standard Scaler	Robust Scaler	No scaling	Standard Scaler	Robust Scaler
5	35.5512	71.6237	71.3079	35.5507	75.9992	76.4258
7	35.8224	73.3797	73.2205	35.8224	80.2579	79.8122
10	36.2383	75.8264	76.4547	36.2378	84.2808	83.5237
12	36.5202	78.0885	78.0719	36.5201	87.2264	85.9928
15	36.9525	79.1931	80.6877	36.9516	89.7520	88.5859
17	37.2456	79.8553	81.4437	37.2455	91.0777	90.0183
19	37.5440	79.3939	82.7809	37.5438	91.7236	91.5733
22	38.0005	80.0719	83.4323	38.0003	93.0081	92.9236
25	38.4687	81.3821	84.3060	38.4681	93.6060	93.6744
27	38.7865	80.9702	84.7790	38.7863	**94.0354**	**94.3113**
30	39.2744	82.0224	84.9896	39.2734	**94.4842**	**94.3097**
32	39.6058	82.5222	**85.0383**	39.6057	**94.6137**	**95.0008**
35	40.1146	**83.3137**	84.5616	40.1139	**95.0970**	**95.1328**
37	40.4605	83.1876	83.2410	40.4606	**95.2350**	**95.4687**
40	40.9915	82.8215	82.5664	40.9909	**95.2086**	**95.6094**

SCENARIO 2 – Encoding the categorical feature

Features

1. SECONDS FROM MIDNIGHT – of the first record in the window
2. ***IS MONDAY – binary feature 0/1***
3. ***IS TUESDAY – binary feature 0/1***
4. ***IS WEDNESDAY – binary feature 0/1***
5. ***IS THURSDAY – binary feature 0/1***
6. ***IS FRIDAY – binary feature 0/1***
7. ***IS SATURDAY – binary feature 0/1***
8. ***IS SUNDAY – binary feature 0/1***
9. SECONDS ELAPSED – between the last and the first record of the window

10.-34. SIMPLE COUNTS OF THE SENSORS

Encoding categorical feature

`sklearn.preprocessing.`**`OneHotEncoder`**

To create correct features which can be processed with SVMs I used **one-of-K** encoding scheme – in Scikit-learn it is `OneHotEncoder`. `OneHotEncoder` transforms each categorical feature with `n_categories` possible values into `n_categories` binary features, with one of them 1, and all others 0. After then feature vectors can feed the estimator.

More on:

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder https://scikit-learn.org/stable/modules/preprocessing.html#encoding-categorical-features

Table 2 – Scenario 2 – Accuracy scores (%)

Window size	Without shuffling			With shuffling		
	No scaling	**Standard Scaler**	**Robust Scaler**	**No scaling**	**Standard Scaler**	**Robust Scaler**
5	35.5512	69.1277	70.4707	35.5507	76.2520	76.7418
7	35.8224	71.3900	73.0136	35.8224	81.0540	80.3694
10	36.2383	74.2643	75.6013	36.2378	84.5385	83.9263
12	36.5202	76.4651	78.2177	36.5201	87.2751	86.6096
15	36.9525	77.3537	80.6055	36.9516	89.8668	89.0785
17	37.2456	77.6200	81.3937	37.2455	91.2763	90.4321
19	37.5440	77.2742	82.3471	37.5438	92.3576	91.8570
22	38.0005	78.2305	82.9422	38.0003	93.1770	93.0756
25	38.4687	80.2364	84.0325	38.4681	93.8111	**94.0334**
27	38.7865	81.1421	**84.8305**	38.7863	**94.0699**	**94.2768**
30	39.2744	81.9347	84.7452	39.2734	**94.4842**	**94.4318**
32	39.6058	82.3633	84.8094	39.6057	**94.7369**	**95.0185**
35	40.1146	**82.8498**	83.7417	40.1139	**94.8474**	**95.2754**
37	40.4605	82.5940	82.9892	40.4606	**95.0551**	**95.6304**
40	40.9915	82.2929	82.6577	40.9909	**95.0446**	**95.9009**

After encoding categorical features accuracy got lower slightly. Specific comparation is at the end of this document.

SCENARIO 3 – Adding PREVIOUS PREDICTED CLASS FEATURE

Features

1. SECONDS FROM MIDNIGHT – of the first record in the window
2. IS MONDAY – binary feature 0/1
3. IS TUESDAY – binary feature 0/1
4. IS WEDNESDAY – binary feature 0/1
5. IS THURSDAY – binary feature 0/1
6. IS FRIDAY – binary feature 0/1
7. IS SATURDAY – binary feature 0/1
8. IS SUNDAY – binary feature 0/1
9. SECONDS ELAPSED – between the last and the first record of the window

10.-34. SIMPLE COUNTS OF THE SENSORS

35. PREVIOUS PREDICTED CLASS WAS 'Phone_Call' – binary feature 0/1

36. PREVIOUS PREDICTED CLASS WAS 'Wash_Hand' – binary feature 0/1

37. PREVIOUS PREDICTED CLASS WAS 'Cook' – binary feature 0/1

38. PREVIOUS PREDICTED CLASS WAS 'Eat' – binary feature 0/1

39. PREVIOUS PREDICTED CLASS WAS 'Clean' – binary feature 0/1

In this scenario I am including PREVIOUS PREDICTED CLASS FEATURE – predicted (computed) class of the anterior feature vector. This feature is categorical. I used **one-of-K** encoding again.

Testing

In this case every feature vector depends on the previous one. Therefore, **we cannot use testing with shuffling the data.**

Feature vectors are tested one by one – with adding the PREVIOUS PREDICTED CLASS FEATURE into the vector which is going to be tested. First vector does not have any preceding feature vector. For this vector is the PREVIOUS PREDICTED CLASS FEATURE calculated as the *mode* (most common value/class) of the training dataset.

Table 3 – Scenario 3 – Accuracy scores (%) without shuffling

Window size	No scaling	Standard Scaler	Robust Scaler
5	35.5512	46.2486	35.5514
7	35.8224	44.7063	42.8114
10	36.2383	44.2269	46.0640
12	36.5202	43.7595	46.4217
15	36.9525	43.0451	48.4331
17	37.2456	43.0394	47.3104
19	37.5440	43.3513	52.7642
22	38.0005	42.6281	57.1873
25	38.4687	43.4950	57.3438
27	38.7865	43.0959	58.2147
30	39.2744	45.0340	62.8041
32	39.6058	45.3788	64.9718
35	40.1146	45.3206	65.4851
37	40.4605	**47.0771**	67.4513
40	**40.9915**	45.7100	**68.9747**

For me it is really surprising that by adding this additional feature the results worsened a lot. I personally expected a significant improvement.

Comparison of the scenarios

In the *Table 4* we are comparing results with **no scaling.** There is no difference between the scenarios. It means that neither encoding ordinal data nor adding extra feature has an influence on the estimator's accuracy.

*Table 4 – Comparison of **NOT SCALED DATA** (accuracy scores %)*

Window size	Without shuffling			With shuffling	
	Scenario 1 **All continuous**	**Scenario 2** **Categorical feat.**	**Scenario 3** **Previous class f.**	**Scenario 1** **All continuous**	**Scenario 2** **Categorical feat.**
5	35.5512	35.5512	35.5512	35.5507	35.5507
7	35.8224	35.8224	35.8224	35.8224	35.8224
10	36.2383	36.2383	36.2383	36.2378	36.2378
12	36.5202	36.5202	36.5202	36.5201	36.5201
15	36.9525	36.9525	36.9525	36.9516	36.9516
17	37.2456	37.2456	37.2456	37.2455	37.2455
19	37.5440	37.5440	37.5440	37.5438	37.5438
22	38.0005	38.0005	38.0005	38.0003	38.0003
25	38.4687	38.4687	38.4687	38.4681	38.4681
27	38.7865	38.7865	38.7865	38.7863	38.7863
30	39.2744	39.2744	39.2744	39.2734	39.2734
32	39.6058	39.6058	39.6058	39.6057	39.6057
35	40.1146	40.1146	40.1146	40.1139	40.1139
37	40.4605	40.4605	40.4605	40.4606	40.4606
40	**40.9915**	**40.9915**	**40.9915**	**40.9909**	**40.9909**

Now let's confront **scaled** data with `StandardScaler` and `RobustScaler` from all three scenarios. Unexpectedly, first scenario gives the best results where I am considering all features to be continuous. Adding PREVIOUS PREDICTED CLASS FEATURE is obviously useless. Is has the worst results from all scenarios.

*Table 5 – Comparison of the results **WITHOUT SHUFFLING** (accuracy score %)*

Window size	Standard Scaler			Robust Scaler		
	Scenario 1 **All continuous**	**Scenario 2** **Categorical feat.**	**Scenario 3** **Previous class f.**	**Scenario 1** **All continuous**	**Scenario 2** **Categorical feat.**	**Scenario 3** **Previous class f.**
5	71.6237	69.1277	46.2486	71.3079	70.4707	35.5514
7	73.3797	71.3900	44.7063	73.2205	73.0136	42.8114
10	75.8264	74.2643	44.2269	76.4547	75.6013	46.0640
12	78.0885	76.4651	43.7595	78.0719	78.2177	46.4217
15	79.1931	77.3537	43.0451	**80.6877**	**80.6055**	48.4331
17	79.8553	77.6200	43.0394	**81.4437**	**81.3937**	47.3104
19	79.3939	77.2742	43.3513	**82.7809**	**82.3471**	52.7642
22	**80.0719**	78.2305	42.6281	**83.4323**	**82.9422**	57.1873
25	**81.3821**	**80.2364**	43.4950	**84.3060**	**84.0325**	57.3438
27	**80.9702**	**81.1421**	43.0959	**84.7790**	**84.8305**	58.2147
30	**82.0224**	**81.9347**	45.0340	**84.9896**	**84.7452**	62.8041
32	**82.5222**	**82.3633**	45.3788	**85.0383**	**84.8094**	64.9718
35	**83.3137**	**82.8498**	45.3206	**84.5616**	**83.7417**	65.4851
37	**83.1876**	**82.5940**	**47.0771**	**83.2410**	**82.9892**	67.4513
40	**82.8215**	**82.2929**	45.7100	**82.5664**	**82.6577**	**68.9747**

In the *Table 6* are compared outcomes from **Scenario 1** and **Scenario 2.** These are the most relevant results because they were tested **with shuffling.** The scores are pretty close – almost the same. In the case of the standardization (Standard Scaler) Scenario 1 (considering all features to be continuous) has a little bit better results. On the other hand, while using Robust Scaler, Scenario 2 (correctly encoding categorical feature) hits better score.

*Table 6 – Comparison of the results **WITH SHUFFLING** (accuracy score %)*

Window size	Standard Scaler		Robust Scaler	
	Scenario 1 All continuous	**Scenario 2** Categorical feat.	**Scenario 1** All continuous	**Scenario 2** Categorical feat.
5	75.9992	76.2520	76.4258	76.7418
7	80.2579	81.0540	79.8122	80.3694
10	84.2808	84.5385	83.5237	83.9263
12	87.2264	87.2751	85.9928	86.6096
15	89.7520	89.8668	88.5859	89.0785
17	91.0777	91.2763	90.0183	90.4321
19	91.7236	92.3576	91.5733	91.8570
22	93.0081	93.1770	92.9236	93.0756
25	93.6060	93.8111	93.6744	**94.0334**
27	**94.0354**	**94.0699**	**94.3113**	**94.2768**
30	**94.4842**	**94.4842**	**94.3097**	**94.4318**
32	**94.6137**	**94.7369**	**95.0008**	**95.0185**
35	**95.0970**	**94.8474**	**95.1328**	**95.2754**
37	**95.2350**	**95.0551**	**95.4687**	**95.6304**
40	**95.2086**	**95.0446**	**95.6094**	**95.9009**

Conclusion

Encoding categorical features does not change the accuracy of the SVMs a lot. Anyway, it is important to create the feature vectors correctly. User guide of Scikit-learn library also suggests to encode all such features using one-of-K encoding scheme before feeding the SVMs.

Adding PREVIOUS PREDICTED CLASS FEATURE does not help at all. Quite the opposite. It decreases the final score significantly.

The highest scores of **more than 95%** are summarized in the *Table 6.* Please, keep in mind that I was using only SVMs classifier with default setting. Thus, I expect even better outcomes after tuning the parameters.

Github

https://github.com/emanuelzaymus/ActivityRecognition