Support Vector Machines – Features 2

Emanuel Zaymus

The objective of this report is to look at the impact of the SECONDS FROM MIDNIGHT OF THE LAST RECORD feature and to try to improve PREVIOUS PREDICTED CLASS feature by filling it with the class probabilities of the previous feature vector.

Dataset

Kyoto – Normal ADL Activities

Scaling

```
sklearn.preprocessing.StandardScaler
```

```
sklearn.preprocessing.RobustScaler
```

Adding SECONDS FROM MIDNIGHT OF THE LAST RECORD feature

Features

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

9.-33. SIMPLE COUNTS OF THE SENSORS

I reduced the count of the day-of-the-week features because I found out that in the dataset there are neither Saturday nor Sunday records.

Used library

```
sklearn.svm.SVC(kernel='rbf', C=1.0, gamma='scale')
```

Testing

```
KFold(n_splits=5, shuffle=True, random_state=0)
```

Adding SECONDS FROM MIDNIGHT – LAST RECORD feature is pretty straightforward. In the *Table 1* we can see the comparison of the scores **with and without** the extra feature. The results are really close. This additional feature caused a slight drop in the accuracy with data scaled using Standard Scaler. On the other hand, the accuracy increased with data scaled using Robust Scaler.

In the upcoming work I will use this feature.

Table 1 – Adding SECONDS FROM MIDNIGHT-LAST RECORD – Accuracy scores (%)

		No scaling		Standard Scaler		Robust Scaler	
w/ SECONDS FROM MIDNIGHT - LAST R.		**No**	**Yes**	**No**	**Yes**	**No**	**Yes**
Window size	5	35.5507	35.5507	76.2520	76.1098	76.7418	76.8366
	7	35.8224	35.8224	81.0540	80.9266	80.3694	80.4650
	10	36.2378	36.2378	84.5385	84.6190	83.9263	84.2324
	12	36.5201	36.5201	87.2751	87.2751	86.6096	86.5446
	15	36.9516	36.9516	89.8668	89.8176	89.0785	89.0786
	17	37.2455	37.2455	91.2763	91.2101	90.4321	90.4652
	19	37.5438	37.5438	92.3576	92.3576	91.8570	91.9238
	22	38.0003	38.0003	93.1770	93.0757	93.0756	93.1939
	25	38.4681	38.4681	93.8111	93.8282	**94.0334**	**94.0676**
	27	38.7863	38.7863	**94.0699**	**94.1906**	**94.2768**	**94.3113**
	30	39.2734	39.2734	**94.4842**	**94.3620**	**94.4318**	**94.4842**
	32	39.6057	39.6057	**94.7369**	**94.6313**	**95.0185**	**95.0360**
	35	40.1139	40.1139	**94.8474**	**94.8474**	**95.2754**	**95.3824**
	37	40.4606	40.4606	**95.0551**	**95.0191**	**95.6304**	**95.6125**
	40	40.9909	40.9909	**95.0446**	**95.0810**	**95.9009**	**95.9738**
	SUM	571.5711	571.5711	1357.8465	1357.3511	1356.6575	1357.6079
	Difference	*0.0000* *No improvement*		***0.4954*** ***Worsening***		***-0.9504*** ***Improvement***	

Improving PREVIOUS PREDICTED CLASS feature

Features

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

9.-33. SIMPLE COUNTS OF THE SENSORS

34. PREVIOUS PREDICTED LIKELIHOOD FOR CLASS 'Phone_Call'

35. PREVIOUS PREDICTED LIKELIHOOD FOR CLASS 'Wash_Hand'

36. PREVIOUS PREDICTED LIKELIHOOD FOR CLASS 'Cook'

37. PREVIOUS PREDICTED LIKELIHOOD FOR CLASS 'Eat'

38. PREVIOUS PREDICTED LIKELIHOOD FOR CLASS 'Clean'

Testing

```
KFold(n_splits=5, shuffle=False)
```

While investigating how to get probabilities from the `sklearn.svm.`**`SVC`** classifier I discovered that there are two different ways to do it:

1. With **`predict_proba`** method of the `SVC` – by getting per-class probabilities
2. With **`decision_function`** of the `SVC` – by getting per-class scores

1. Using `predict_proba` method

Classifier setup

```
sklearn.svm.SVC(probability=True, break_ties=True)
```

This method returns per-class probabilities for a given feature vector, which is exactly what we want. The only downside is that it comes with **expensive computational costs**.

In the next table we are going to compare the accuracies with the results from the last report, where we treated PREVIOUS PREDICTED CLASS feature as binary feature.

Table 2 – PREVIOUS PREDICTED CLASS using `predict_proba` – Accuracy scores (%)

Window size	No scaling		Standard Scaler		Robust Scaler	
	binary feature	predict_proba	binary feature	predict_proba	binary feature	predict_proba
5	35.5512	35.5512	46.2486	53.7848	35.5514	36.3574
7	35.8224	35.8224	44.7063	54.0193	42.8114	42.4312
10	36.2383	36.2383	44.2269	55.6939	46.0640	50.3802
12	36.5202	36.5202	43.7595	54.8276	46.4217	53.5001
15	36.9525	36.9525	43.0451	57.2195	48.4331	54.6587
17	37.2456	37.2456	43.0394	58.2178	47.3104	59.2312
19	37.5440	37.5440	43.3513	57.6190	52.7642	56.7537
22	38.0005	38.0005	42.6281	58.3847	57.1873	55.9898
25	38.4687	38.4687	43.4950	58.8321	57.3438	64.5595
27	38.7865	38.7865	43.0959	60.5923	58.2147	65.4387
30	39.2744	39.2744	45.0340	64.1841	62.8041	70.0660
32	39.6058	39.6058	45.3788	64.6542	64.9718	71.5725
35	40.1146	40.1146	45.3206	63.9351	65.4851	74.8267
37	40.4605	40.4605	47.0771	**64.9523**	67.4513	76.1739
40	40.9915	40.9915	45.7100	62.4357	68.9747	**77.6473**

As we can see, the results improved, but it is still not what I would expect from adding such a feature.

Documentation for `predict_proba` metod:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC.predict_proba

2. Using `decision_function`

Classifier setup

```
sklearn.svm.SVC()
```

Decision function of the SVC does not return probabilities but confidence score for every class. It does not need to necessarily be in a range between 0 and 1 as we expect when dealing with probabilities. So, I needed to **change the training data**.

In the previous case I set *PREVIOUS PREDICTED LIKELIHOOD FOR CLASS 'x'* in the training data to 1 when the previous class was 'x' otherwise 0. I am setting it to value of 1 because it is the maximum what can likelihood obtain.

I decided to replace the value of 1 with maximum value what `decision_function` can return. Maximum value which is given by `decision_function` for this dataset is around 4.2. Because of that I decided to use value of 5.

Now let's take a look at the *Table 3* where are the results.

Table 3 – PREVIOUS PREDICTED CLASS using `decision_function` *– Accuracy scores (%)*

Window size	No scaling		Standard Scaler		Robust Scaler	
	binary feature	decision_func	binary feature	decision_func	binary feature	decision_func
5	35.5512	35.5512	46.2486	35.1724	35.5514	24.7119
7	35.8224	35.8224	44.7063	37.6051	42.8114	28.4827
10	36.2383	36.2383	44.2269	40.5717	46.0640	31.5846
12	36.5202	36.5202	43.7595	41.5998	46.4217	33.5009
15	36.9525	36.9525	43.0451	43.9991	48.4331	34.2439
17	37.2456	37.2456	43.0394	44.2472	47.3104	35.1599
19	37.5440	37.5440	43.3513	44.5220	52.7642	35.8621
22	38.0005	38.0005	42.6281	46.7315	57.1873	36.6667
25	38.4687	38.4687	43.4950	48.1989	57.3438	35.5461
27	38.7865	38.7865	43.0959	50.2840	58.2147	34.6833
30	39.2744	39.2744	45.0340	50.6921	62.8041	37.6694
32	39.6058	39.6058	45.3788	52.1034	64.9718	37.9504
35	40.1146	40.1146	45.3206	51.1884	65.4851	40.4008
37	40.4605	40.4605	47.0771	51.6272	67.4513	41.0348
40	40.9915	40.9915	45.7100	**52.3619**	**68.9747**	43.1611

Here we can see that such a feature improves only data scaled with Standard Scaler. Robust scaled data reached very poor results.

Documentation for `decision_funtion`:
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC.decision_function

More about Scores and Probabilities on:
https://scikit-learn.org/stable/modules/svm.html#scores-probabilities

Comparison

Table 4 – PREVIOUS PREDICTED CLASS Comparison – Accuracy scores (%)

Window size	Standard Scaler			Robust Scaler		
	binary feature	predict_proba	decision_func	binary feature	predict_proba	decision_func
5	46.2486	53.7848	35.1724	35.5514	36.3574	24.7119
7	44.7063	54.0193	37.6051	42.8114	42.4312	28.4827
10	44.2269	55.6939	40.5717	46.0640	50.3802	31.5846
12	43.7595	54.8276	41.5998	46.4217	53.5001	33.5009
15	43.0451	57.2195	43.9991	48.4331	54.6587	34.2439
17	43.0394	58.2178	44.2472	47.3104	59.2312	35.1599
19	43.3513	57.6190	44.5220	52.7642	56.7537	35.8621
22	42.6281	58.3847	46.7315	57.1873	55.9898	36.6667
25	43.4950	58.8321	48.1989	57.3438	64.5595	35.5461
27	43.0959	60.5923	50.2840	58.2147	65.4387	34.6833
30	45.0340	64.1841	50.6921	62.8041	70.0660	37.6694
32	45.3788	64.6542	52.1034	64.9718	71.5725	37.9504
35	45.3206	63.9351	51.1884	65.4851	74.8267	40.4008
37	47.0771	**64.9523**	51.6272	67.4513	76.1739	41.0348
40	45.7100	62.4357	52.3619	68.9747	**77.6473**	43.1611

Conclusion

PREVIOUS PREDICTED CLASS feature does not seem to add any value to our problem. Even though I tried three versions of this feature (binary, likelihood, confidence score) it worsened the accuracy every time.

For this reason, I do not suggest to include this feature into the final feature setup.

Github

https://github.com/emanuelzaymus/ActivityRecognition