3 Variable Stars (36 points)

A variable star changes its intensity, as observed by a telescope, over time. This can be caused extrinsically, for example by other objects temporarily occluding it, but also intrinsically, when the star changes its physical properties over time. Figure 1 shows an example. The graph of the varying intensity as a function of time is called the light curve. Variable stars can be further divided into many classes depending on other physical properties. The task we are trying to solve is to predict the class of a variable star by its light curve. To achieve this, we train a classifier in a supervised setting using labeled data from the All Sky Automated Survey Catalog of Variable Stars (ACVS) [Pojmanski, 2000].

The data considered in the following is based on the study by Richards et al. [2012]. We have a training and a test set, in the file `VSTrain.dt` and `VSTest.dt`, respectively, with 771 labeled samples each. Each sample encodes the astronomical properties of a variable star in a 61-dimensional feature vector. The features are listed in Table 3, for a detailed description of their meaning we refer to Dubath et al. [2011] and Richards et al. [2011]. The labels indicate the class a variable star has been assigned to. In total there are 25 different classes, see Table 3.

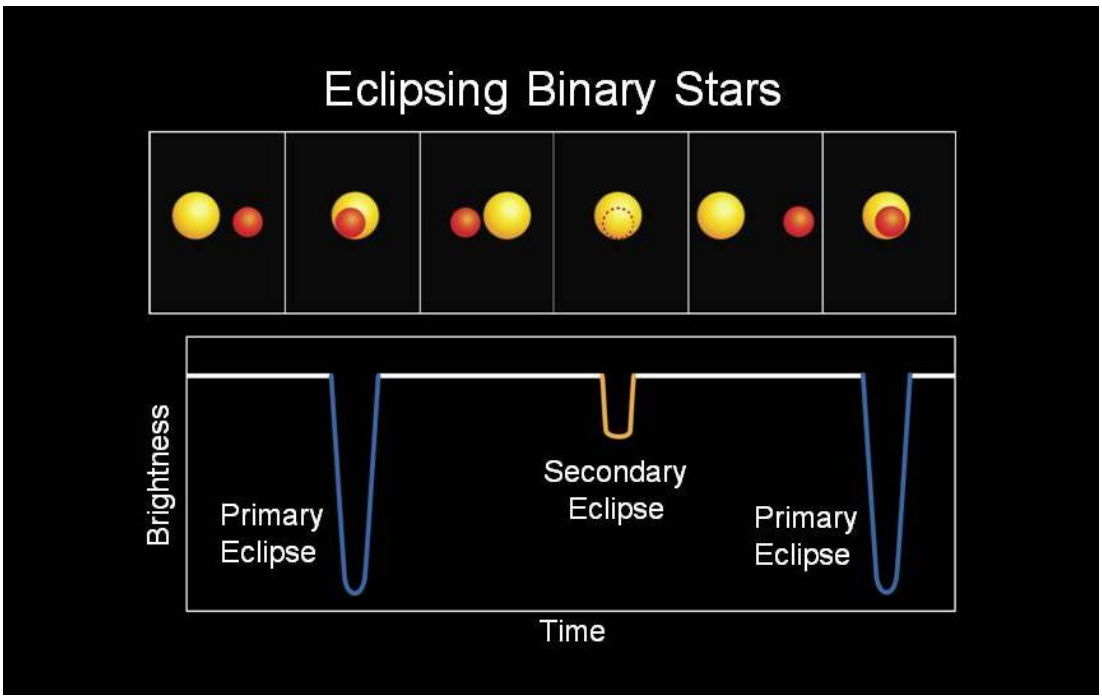

Figure 1: A variable star changes its intensity as observed from a telescope due to another smaller orbiting star. The image is taken from the NASA, `http://kepler.nasa.gov/news/nasakeplernews/index.cfm?FuseAction=ShowNews&NewsID=152`.

#	Feature name	#	Feature name
0	amplitude	31	freq3_harmonics_rel_phase_2
1	beyond1std	32	freq3_harmonics_rel_phase_3
2	flux_percentile_ratio_mid20	33	freq_amplitude_ratio_21
3	flux_percentile_ratio_mid35	34	freq_amplitude_ratio_31
4	flux_percentile_ratio_mid50	35	freq_frequency_ratio_21
5	flux_percentile_ratio_mid65	36	freq_frequency_ratio_31
6	flux_percentile_ratio_mid80	37	freq_signif
7	fold2P_slope_10percentile	38	freq_signif_ratio_21
8	fold2P_slope_90percentile	39	freq_signif_ratio_31
9	freq1_harmonics_amplitude_0	40	freq_varrat
10	freq1_harmonics_amplitude_1	41	freq_y_offset
11	freq1_harmonics_amplitude_2	42	linear_trend
12	freq1_harmonics_amplitude_3	43	max_slope
13	freq1_harmonics_freq_0	44	median_absolute_deviation
14	freq1_harmonics_rel_phase_1	45	median_buffer_range_percentage
15	freq1_harmonics_rel_phase_2	46	medperc90_2p_p
16	freq1_harmonics_rel_phase_3	47	p2p_scatter_2praw
17	freq2_harmonics_amplitude_0	48	p2p_scatter_over_mad
18	freq2_harmonics_amplitude_1	49	p2p_scatter_pfold_over_mad
19	freq2_harmonics_amplitude_2	50	p2p_ssqr_diff_over_var
20	freq2_harmonics_amplitude_3	51	percent_amplitude
21	freq2_harmonics_freq_0	52	percent_difference_flux_percentile
22	freq2_harmonics_rel_phase_1	53	QSO
23	freq2_harmonics_rel_phase_2	54	non_QSO
24	freq2_harmonics_rel_phase_3	55	scatter_res_raw
25	freq3_harmonics_amplitude_0	56	skew
26	freq3_harmonics_amplitude_1	57	small_kurtosis
27	freq3_harmonics_amplitude_2	58	std
28	freq3_harmonics_amplitude_3	59	stetson_j
29	freq3_harmonics_freq_0	60	stetson_k
30	freq3_harmonics_rel_phase_1		

Table 1: Different features are used to describe the light curve of a variable star.

3.1 Data understanding and preprocessing

Download the data in `VSTrain.dt` and `VSTest.dt`. Each line contains the input and as last value in a row the target label.

Report the class frequencies, that is, for each class report the number of data points belonging to that class divided by the total number of data points in the training data.

Label	Class name	Label	Class name
0	Mira	13	Gamma Doradus
1	Semireg PV	14	Pulsating Be
2	RV Tauri	15	Per. Var. SG
3	Classical Cep	16	Chem. Peculia
4	Pop. II Cephe	17	Wolf-Rayet
5	Multi. Mode C	18	T Tauri
6	RR Lyrae, FM	19	Herbig AE/BE
7	RR Lyrae, FO	20	S Doradus
8	RR Lyrae, DM	21	Ellipsoidal
9	Delta Scuti	22	Beta Persei
10	Lambda Bootis	23	Beta Lyrae
11	Beta Cephei	24	W Ursae Maj
12	Slowly Puls.		

Table 2: The 25 different classes a variable star can be assigned to.

Then conduct two preprocessing steps.

1. Remove all data points belonging to classes with less than 65 training examples. Report which classes and how many training and test examples remain.

   Hint: Assuming `import numpy as np`, the Python functions `np.unique(..., return_counts=True)`, `np.where(...)`, and `np.delete(...)` (with `axis=0` on the input data) may be useful.

2. Normalize the data to zero mean and unit variance on the training data set.

Deliverables: frequency of classes in original; number of classes and number of training and test examples after removing small classes; code snippets for the normalization and data removal in the report

3.2 Principal component analysis

Perform a principal component analysis (PCA) of the normalized training data.[1] Visualize the data by a scatter plot of the data projected on the first two principal components. Use different colors for the different classes in the plot so that the points belonging to different classes can be clearly distinguished.

Deliverables: scatter plot of the data projected on the first two principal components with different colors indicating the different classes

[1]Note that PCA results including the eigenspectrum change due to the normalization.

3.3 Clustering

Perform 4-means clustering of the training data.

After that, project the cluster centers to the first two principal components of the training data. Then visualize the clusters by adding the cluster centers to the plot from the previous exercise.

Briefly discuss the results.

Deliverables: description of software used; one plot with cluster centers and data points; short discussion of results

3.4 Classification

The task is to evaluate several multi-class classifiers on the data. Build the models using the training data only. The test data must only be used for final evaluation.

1. Apply multi-nominal logistic regression. If you use regularization, describe the type of regularization you used. Report training and test loss (in terms of 0-1 loss).

2. Apply random forests with 200 trees. In one setting, set the number of features considered when looking for the best split to the square root of the total number of features. In a second setting, set the number of features considered when looking for the best split to the total number of features. Report training and test loss (in terms of 0-1 loss) and the out-of-bag (OOB) *error*.

3. Apply k-nearest-neighbor classification. Use cross-validation to determine the number of neighbors. Report training and test loss (in terms of 0-1 loss). Describe how you determined the number of neighbors.

Deliverables: description of software used; training and test errors; OOB errors for the random forests; description of regularization and model selection process

~~4 Theoretically justified cross-validation (20 points)~~

~~Cross-validation is arguably the most widely-used method for parameter tuning in machine learning applications. The standard cross-validation procedure prescribes to cross-validate the parameters using cross-validation split of the data, select the best parameter [according to the cross-validation error], and then train~~