CS 181 Practical

Names: Ethan Lee, Leonard Tang
Emails: ethan_lee@college.harvard.edu, leonardtang@college.harvard.edu

May 7, 2022

1 Part A: Feature Engineering, Baseline Models

1.1 Approach

As discussed in class, the top 500 principal components represent the 500 most salient directions of the data, i.e. the directions of the data capturing the maximal amount of variance. On a high level, PCA is performed by minimizing the reconstruction error, or the error from reconstructing the data using the principal components that have been calculated. Looking more deeply at the mathematical operations taking place, PCA is conducted by calculating the eigenvectors of the sample covariance, and omitting the eigenvectors with the smallest corresponding eigenvalues. By doing this, we can minimize the reconstruction error, and we can easily calculate the eigenvectors using singular value decomposition as the sample covariance is symmetric.

Regarding logistic regression, we used the scikit-learn package implementing it, which uses a gradient descent to constantly minimize the negative log-likelihood loss and find the optimal weight values (to be applied to each data point) that do so, such that once the softmax function is applied to the entire weighted dataset using the optimal weights, we can output probabilities that each data point is in each specific class.

1.2 Results

Model	Train Ovr	Test Ovr	0	1	2	3	4	5	6	7	8	9
AMP	41.17	18.57	22.67	2.56	54.52	7.42	8.33	14.39	6.67	10.17	11.02	15.67
MEL	69.82	31.41	19.67	56.41	54.18	18.78	32.95	34.09	60	22.03	45.34	16.67

Table 1: Accuracies (%) for Log. Regression Using Raw Amplitude and Mel Spectrogram Data

The overall and test per-class accuracy percentages for logistic regression when using the principal components of the amplitude data (Baseline 1) and Mel spectrogram data (Baseline 2) are shown above (full train per-class accuracies are in appendix); logistic regression using the amplitude data led to overall test accuracy 18.57%, and logistic regression using the Mel spectrogram data led to overall test accuracy 31.41%. Some interesting results can be noted; in particular, notice that for both models, sounds with consistent levels of loudness and frequency (such as Class 2:

children playing) exhibited high relative accuracy for both models. Obviously, this is just an initial observation and not a heuristic that holds universally.

1.3 Discussion

We can see that the logistic regression model trained on Mel Spectrogram data performed better, which would make sense because the Mel Spectrogram data interprets the frequencies/pitches of the sounds while the raw amplitude simply takes the "loudness" into account; furthermore, many of these sounds could have very varying amplitudes, but would have relatively consistent frequencies (especially categories such as car horn, gun shot, and siren, which had the largest differences in accuracy between the two models).

We were most likely asked to perform PCA first due to the vast number of features available which we would have wanted to reduce, and the impact of performing PCA first was that the training of the logistic regression models took much less time than it could have, and also that only the 500 most important features (from tens of thousands of features) were extracted, allowing for us to train models based only on the most crucial components of the dataset. Furthermore, it is standard practice to perform PCA first when analyzing a dataset in order to get a better understanding of the most impactful features and whether dimensionality reduction would be feasible.

2 Part B: More Modeling

2.1 First Step

2.1.1 Approach

For this step, we used a SVM with the non-linear radial basis function as a kernel on the Mel spectrogram PC's, as implemented in the scikit-learn ensemble. Mathematically, SVM with this kernel works by first redefining the optimization problem in SVM so that we can work in higher dimensions; the new objective function is $\min_{w,w_0}(\max_{\alpha_n} \frac{1}{2}w^T w - \sum_n \alpha_n(y_n(w^T x_n + w_0) - 1))$, where w are our weights and $\alpha_n \geq 0$. Then, using strong duality, one can switch the minimization and maximization parts of this objective function and rearrange terms, such that the objective becomes $\max_\alpha -\frac{1}{2}\sum_n \sum_{n'} \alpha_n \alpha'_n y_n y'_n x_n x'_n + \sum \alpha_n$, where $\alpha_n \geq 0$ and $\sum_n \alpha_n y_n = 0$. Applying the kernel trick, one can then replace the instances of x_n and x'_n with the radial basis function as a kernel, which is $\exp(-\frac{||x-x'||^2}{2\sigma^2})$ for two sample data points x and x', where σ is a free parameter. We then solve for α_n values using standard optimization techniques such as stochastic gradient descent.

2.1.2 Results

Model	Train Ovr	Test Ovr	0	1	2	3	4	5	6	7	8	9
SVM	44.61	29.18	28.67	43.59	88.96	17.90	16.29	6.44	20	11.44	33.05	20

Table 2: SVM Accuracy (%) Using Mel Spectrogram Data

The overall and test per-class classification accuracy of the SVM model are shown in the table above (full train per-class accuracies are in appendix). We can see that the SVM model had an

overall test accuracy of 29.18%, with its best performance coming in identifying samples from class 2 (children playing), with 88.96% accuracy.

2.1.3 Discussion

Overall, the results achieved by the SVM model were similar but slightly worse than the results using logistic regression (overall accuracy of 29.18% compared to 31.41%), which would suggest that a non-linear approach would not always be best when dealing with this specific classification problem. With regards to SVM specifically, restructuring data samples from classes that already had very high variance in frequency may not have had much of an effect on making them more separable (as can be seen in the per-class accuracies), and the radial basis function as a kernel may not have been effective in this regard.

Some observations to be noted are that the SVM model performed very badly on classes 5 and 7 (while the logistic regression model didn't perform as badly), and this may be explained by the fact that these classes especially (engine idling and jackhammer) will have sounds of extremely varying frequencies occur very close to each other over time, and so the SVM model which restructures data based on distances from other points would have a hard time classifying them. The SVM model performed exceptionally well on class 2, which might mean that the sounds of children playing might be more "recognizable", or separable from data samples of other classes, after being transformed through a kernel function such as the radial basis function; this might make sense as children playing would have a very unique frequency when compared to the other sounds in this dataset. Overall, the results from the non-linear SVM show the complexity of this problem, as the presence of classes of different consistencies and frequencies might lead to difficulty in one method being able to correctly identify all of them.

2.2 Hyperparameter Tuning and Validation

For this section, we used the 500 most important principal components of the Mel spectrogram data. Our first approach for an additional nonlinear model was a random forest model, with hyperparameter tuning conducted through cross-validation. As a simple mathematical explanation behind the functions of a random forest, a random forest utilizes the wisdom of crowds approach. Through bagging, a random forest model will generate a certain number of decision trees, each of which are trained on a different sample (picked with replacement) of the original dataset and each of which can only access a certain random subset of the dataset's features. The random forest then uses the average result of each of the trees to come to its prediction.

For hyperparameter search, we used the standard GridSearchCV method in scikit-learn that uses a 5-fold cross validation by default. Cross validation works by splitting the training dataset into k segments, and then using each of those segments as a validation set once each while using the other $k-1$ segments as training data. As a result, the cross-validation approach tests which hyperparameter combinations for a model perform the best without having to access the test data.

For random forest, the hyperparameters we tested were the number of trees in each random forest (100, 200, 500), the maximum depth which is the number of splits each tree can have (3, 5,

10, 15, 25), and the minimum samples split at each decision tree node (5, 10, 15).

We also elected to use SVM (again using a RBF kernel) as our second model class. Critically, we search over the regularization parameter C with candidate values (0.1, 1, 3, 5, 7, 10) and the kernel coefficient γ with candidate values (1, 0.1, 0.01, 0.001, 0.0001).

2.2.1 Results

Model	Train Ovr	Test Ovr	0	1	2	3	4	5	6	7	8	9
RF	100	40.78	18	17.95	59.53	25.33	33.33	33.71	10	56.36	66.10	43.33
SVM	85.02	31.41	15	2.56	64.21	4.37	22.73	5.68	6.67	49.15	14.83	71.33

Table 3: RF and SVM Accuracy (%) Using Mel Spectrogram Data

The full results of our hyperparameter searches are contained in the tables in the appendix, and the overall and test per-class accuracies achieved by the models when using the optimal hyperparameters are shown in the table above (full train per-class accuracies are in appendix). The optimal hyperparameters for random forest were `n_estimators = 500`, `maxdepth = 50`, `min_samples_split = 3`, and for SVM were `C = 5`, γ `= 0.0001`. Modeled on the test dataset, these combinations of hyperparameters led to an overall test accuracy of 40.78% for random forest and 31.41% for SVM.

2.2.2 Discussion

As discussed when explaining the approaches used in this part, the validation strategy used was 5-fold cross-validation, which should be effective at identifying the optimal hyperparameters as it runs 5 "trials" of designating a validation set and testing hyperparameters' performances on them without accessing the test data. The tuned models were expected to perform better than the baseline models firstly because tuning the non-linear models allows for a unique form of separation of the data samples that could improve classification performance; SVM finds the best margin of separation for data samples and random forest bins similar data samples together, compared to logistic regression which uses simpler weight-based separation techniques. Additionally, logistic regression is more prone to overfitting. Regarding the First Step model, for SVM specifically, we used the same model class as the First Step, but with untuned parameters judged purely on intuition; thus, it only makes sense that the SVM model with the tuned parameters would perform better than the one in the First Step, as it uses the optimal hyperparameters (at least from the ones we chose to test). For random forest specifically, the model uses an approach that is intrinsically suited for multiclass problems, while SVM is typically best used for the two-class case. Additionally, the presence of classes with very small frequency and outliers in the dataset would have improved random forest's performance compared to other models, since it can essentially put outliers into their own bin compared to other models whose weights might be affected by outliers. As some general conclusions, we were able to generate a model with at least 25% accuracy using the logistic regression model, SVM model, and tuned models on Mel spectrogram PC's. Overall, our results from this practical show the complexity in correctly classifying data samples of varying structures (i.e. the frequencies and amplitudes within this specific dataset), as some models will do better at identifying data samples of certain classes.

3 Appendix

3.1 Train Set Per-Class Accuracies

Model	Train Ovr	0	1	2	3	4	5	6	7	8	9
AMP LR	41.17	40.86	34.52	56.03	30.40	32.13	45.08	63.86	40.27	40.42	39.29
MEL LR	69.82	62.29	84.26	70.69	69.79	69.85	67.27	91.57	73.56	71.86	67
SVM	44.61	41.71	25.38	91.52	31.93	26.19	37.03	73.49	57.29	32.34	35.86
RF	100	100	100	100	100	100	100	100	100	100	100
SVM Tuned	85.02	80.29	85.28	96.55	76.48	86.99	84.60	91.57	93.16	75.45	84

Table 4: Training Set Per-Class Accuracies

3.2 Random Forest Hyperparameter Tuning Results

	param_n_estimators	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score
50	500	{'max_depth': 50, 'min_samples_split': 3, 'n_e...	0.395140	0.480648	0.429343	0.454054	0.373874	0.426612
38	500	{'max_depth': 25, 'min_samples_split': 3, 'n_e...	0.395140	0.479748	0.426643	0.455856	0.373874	0.426252
53	500	{'max_depth': 50, 'min_samples_split': 5, 'n_e...	0.399640	0.478848	0.423942	0.448649	0.363964	0.423009
41	500	{'max_depth': 25, 'min_samples_split': 5, 'n_e...	0.399640	0.477948	0.424842	0.448649	0.363964	0.423009
52	200	{'max_depth': 50, 'min_samples_split': 5, 'n_e...	0.396940	0.468047	0.422142	0.448649	0.359459	0.419047
40	200	{'max_depth': 25, 'min_samples_split': 5, 'n_e...	0.396940	0.468047	0.422142	0.447748	0.359459	0.418867
56	500	{'max_depth': 50, 'min_samples_split': 10, 'n_...	0.383438	0.464446	0.421242	0.450450	0.362162	0.416348
44	500	{'max_depth': 25, 'min_samples_split': 10, 'n_...	0.383438	0.464446	0.421242	0.449550	0.362162	0.416168
37	200	{'max_depth': 25, 'min_samples_split': 3, 'n_e...	0.377138	0.454545	0.422142	0.460360	0.354955	0.413828
49	200	{'max_depth': 50, 'min_samples_split': 3, 'n_e...	0.377138	0.455446	0.421242	0.458559	0.354955	0.413468
47	500	{'max_depth': 25, 'min_samples_split': 15, 'n_...	0.388839	0.458146	0.405041	0.424324	0.375676	0.410405
59	500	{'max_depth': 50, 'min_samples_split': 15, 'n_...	0.389739	0.458146	0.405041	0.423423	0.375676	0.410405
43	200	{'max_depth': 25, 'min_samples_split': 10, 'n_...	0.376238	0.474347	0.403240	0.429730	0.354054	0.407522
55	200	{'max_depth': 50, 'min_samples_split': 10, 'n_...	0.376238	0.474347	0.403240	0.427928	0.354054	0.407161
46	200	{'max_depth': 25, 'min_samples_split': 15, 'n_...	0.385239	0.450045	0.404140	0.422523	0.367568	0.405903
58	200	{'max_depth': 50, 'min_samples_split': 15, 'n_...	0.385239	0.450045	0.404140	0.422523	0.367568	0.405903
39	100	{'max_depth': 25, 'min_samples_split': 5, 'n_e...	0.388839	0.447345	0.417642	0.429730	0.345946	0.405900
51	100	{'max_depth': 50, 'min_samples_split': 5, 'n_e...	0.388839	0.447345	0.417642	0.429730	0.345946	0.405900
57	100	{'max_depth': 50, 'min_samples_split': 15, 'n_...	0.381638	0.437444	0.405041	0.420721	0.372072	0.403383
45	100	{'max_depth': 25, 'min_samples_split': 15, 'n_...	0.381638	0.437444	0.405041	0.419820	0.372072	0.403203
28	200	{'max_depth': 10, 'min_samples_split': 5, 'n_e...	0.378938	0.440144	0.405041	0.436937	0.347748	0.401761
29	500	{'max_depth': 10, 'min_samples_split': 5, 'n_e...	0.388839	0.440144	0.394239	0.421622	0.363063	0.401581
36	100	{'max_depth': 25, 'min_samples_split': 3, 'n_e...	0.369037	0.441044	0.416742	0.427027	0.351351	0.401040
48	100	{'max_depth': 50, 'min_samples_split': 3, 'n_e...	0.369037	0.438344	0.414941	0.429730	0.351351	0.400681
26	500	{'max_depth': 10, 'min_samples_split': 3, 'n_e...	0.384338	0.432943	0.387039	0.423423	0.366667	0.398882
25	200	{'max_depth': 10, 'min_samples_split': 3, 'n_e...	0.366337	0.447345	0.394239	0.427027	0.354955	0.397981
42	100	{'max_depth': 25, 'min_samples_split': 10, 'n_...	0.369937	0.442844	0.401440	0.419820	0.339640	0.394736
54	100	{'max_depth': 50, 'min_samples_split': 10, 'n_...	0.369937	0.442844	0.401440	0.419820	0.339640	0.394736
32	500	{'max_depth': 10, 'min_samples_split': 10, 'n_...	0.386139	0.438344	0.381638	0.404505	0.356757	0.393476
31	200	{'max_depth': 10, 'min_samples_split': 10, 'n_...	0.378038	0.424842	0.387039	0.414414	0.354054	0.391677
24	100	{'max_depth': 10, 'min_samples_split': 3, 'n_e...	0.376238	0.427543	0.381638	0.418018	0.345045	0.389696
27	100	{'max_depth': 10, 'min_samples_split': 5, 'n_e...	0.369037	0.412241	0.393339	0.412613	0.360360	0.389518
35	500	{'max_depth': 10, 'min_samples_split': 15, 'n_...	0.375338	0.429343	0.383438	0.402703	0.354955	0.389155
34	200	{'max_depth': 10, 'min_samples_split': 15, 'n_...	0.386139	0.426643	0.370837	0.392793	0.350450	0.385372
30	100	{'max_depth': 10, 'min_samples_split': 10, 'n_...	0.371737	0.417642	0.379838	0.404505	0.337838	0.382312
33	100	{'max_depth': 10, 'min_samples_split': 15, 'n_...	0.370837	0.413141	0.369937	0.389189	0.353153	0.379252
20	500	{'max_depth': 5, 'min_samples_split': 10, 'n_e...	0.327633	0.343834	0.288929	0.363964	0.318919	0.328656
12	100	{'max_depth': 5, 'min_samples_split': 3, 'n_es...	0.329433	0.351935	0.279028	0.352252	0.328829	0.328295
19	200	{'max_depth': 5, 'min_samples_split': 10, 'n_e...	0.332133	0.342934	0.280828	0.362162	0.318919	0.327395
17	500	{'max_depth': 5, 'min_samples_split': 5, 'n_es...	0.328533	0.340234	0.283528	0.367568	0.314414	0.326855
22	200	{'max_depth': 5, 'min_samples_split': 15, 'n_e...	0.333933	0.342034	0.283528	0.362162	0.312613	0.326854
16	200	{'max_depth': 5, 'min_samples_split': 5, 'n_es...	0.329433	0.336634	0.279928	0.363964	0.323423	0.326676
15	100	{'max_depth': 5, 'min_samples_split': 5, 'n_es...	0.331233	0.347435	0.275428	0.350450	0.328829	0.326675
23	500	{'max_depth': 5, 'min_samples_split': 15, 'n_e...	0.329433	0.354635	0.280828	0.363063	0.304505	0.326493
13	200	{'max_depth': 5, 'min_samples_split': 3, 'n_es...	0.327633	0.334833	0.278128	0.365766	0.320721	0.325416
18	100	{'max_depth': 5, 'min_samples_split': 10, 'n_e...	0.338434	0.341134	0.279928	0.354955	0.312613	0.325413
14	500	{'max_depth': 5, 'min_samples_split': 3, 'n_es...	0.328533	0.340234	0.278128	0.365766	0.310811	0.324694
21	100	{'max_depth': 5, 'min_samples_split': 15, 'n_e...	0.331233	0.347435	0.285329	0.347748	0.311712	0.324691
11	500	{'max_depth': 3, 'min_samples_split': 15, 'n_e...	0.326733	0.305131	0.252025	0.330631	0.286486	0.300201
8	500	{'max_depth': 3, 'min_samples_split': 10, 'n_e...	0.326733	0.305131	0.251125	0.330631	0.286486	0.300021
5	500	{'max_depth': 3, 'min_samples_split': 5, 'n_es...	0.326733	0.305131	0.251125	0.330631	0.286486	0.300021
2	500	{'max_depth': 3, 'min_samples_split': 3, 'n_es...	0.326733	0.305131	0.251125	0.330631	0.286486	0.300021
10	200	{'max_depth': 3, 'min_samples_split': 15, 'n_e...	0.314131	0.306931	0.250225	0.318018	0.285586	0.294978
1	200	{'max_depth': 3, 'min_samples_split': 3, 'n_es...	0.314131	0.306931	0.251125	0.318919	0.282883	0.294798
4	200	{'max_depth': 3, 'min_samples_split': 5, 'n_es...	0.314131	0.306931	0.251125	0.318919	0.282883	0.294798
7	200	{'max_depth': 3, 'min_samples_split': 10, 'n_e...	0.314131	0.306931	0.250225	0.318018	0.282883	0.294438
0	100	{'max_depth': 3, 'min_samples_split': 3, 'n_es...	0.306031	0.297930	0.258326	0.310811	0.288288	0.292277
3	100	{'max_depth': 3, 'min_samples_split': 5, 'n_es...	0.306031	0.297930	0.258326	0.310811	0.288288	0.292277
6	100	{'max_depth': 3, 'min_samples_split': 10, 'n_e...	0.306031	0.297930	0.257426	0.310811	0.288288	0.292097
9	100	{'max_depth': 3, 'min_samples_split': 15, 'n_e...	0.306031	0.297930	0.257426	0.310811	0.286486	0.291737

3.3 SVM Hyperparameter Tuning Results

	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
19	{'C': 5, 'gamma': 0.0001, 'kernel': 'rbf'}	0.240324	0.315032	0.272727	0.290991	0.239640	0.271743	0.029200	1
24	{'C': 7, 'gamma': 0.0001, 'kernel': 'rbf'}	0.237624	0.308731	0.268227	0.296396	0.237838	0.269763	0.029265	2
29	{'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}	0.234023	0.306031	0.269127	0.293694	0.244144	0.269404	0.027646	3
14	{'C': 3, 'gamma': 0.0001, 'kernel': 'rbf'}	0.235824	0.315932	0.260126	0.289189	0.236937	0.267601	0.031009	4
9	{'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}	0.219622	0.294329	0.241224	0.271171	0.226126	0.250495	0.028220	5
18	{'C': 5, 'gamma': 0.001, 'kernel': 'rbf'}	0.199820	0.211521	0.190819	0.150450	0.214414	0.193405	0.023076	6
23	{'C': 7, 'gamma': 0.001, 'kernel': 'rbf'}	0.201620	0.209721	0.187219	0.152252	0.211712	0.192505	0.021893	7
13	{'C': 3, 'gamma': 0.001, 'kernel': 'rbf'}	0.195320	0.210621	0.188119	0.154955	0.209910	0.191785	0.020326	8
28	{'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}	0.201620	0.205221	0.185419	0.151351	0.208108	0.190344	0.021018	9
8	{'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}	0.192619	0.206121	0.196220	0.160360	0.195495	0.190163	0.015582	10
4	{'C': 0.1, 'gamma': 0.0001, 'kernel': 'rbf'}	0.162016	0.189919	0.163816	0.145045	0.181982	0.168556	0.015841	11
27	{'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}	0.195320	0.159316	0.152115	0.136036	0.178378	0.164233	0.020652	12
22	{'C': 7, 'gamma': 0.01, 'kernel': 'rbf'}	0.193519	0.157516	0.152115	0.135135	0.180180	0.163693	0.020733	13
17	{'C': 5, 'gamma': 0.01, 'kernel': 'rbf'}	0.193519	0.158416	0.146715	0.129730	0.180180	0.161712	0.022854	14
12	{'C': 3, 'gamma': 0.01, 'kernel': 'rbf'}	0.198920	0.159316	0.144914	0.120721	0.179279	0.160630	0.027038	15
11	{'C': 3, 'gamma': 0.1, 'kernel': 'rbf'}	0.168317	0.141314	0.154815	0.170270	0.158559	0.158655	0.010428	16
26	{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}	0.169217	0.139514	0.153915	0.170270	0.154054	0.157394	0.011389	17
16	{'C': 5, 'gamma': 0.1, 'kernel': 'rbf'}	0.168317	0.139514	0.153915	0.168468	0.155856	0.157214	0.010733	18
21	{'C': 7, 'gamma': 0.1, 'kernel': 'rbf'}	0.168317	0.139514	0.152115	0.169369	0.152252	0.156314	0.011233	19
6	{'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}	0.164716	0.139514	0.139514	0.169369	0.155856	0.153794	0.012442	20
7	{'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}	0.179118	0.145815	0.138614	0.118018	0.181982	0.152709	0.024511	21
2	{'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}	0.170117	0.127813	0.130513	0.159459	0.154955	0.148571	0.016616	22
15	{'C': 5, 'gamma': 1, 'kernel': 'rbf'}	0.153015	0.135014	0.153015	0.145045	0.144144	0.146047	0.006685	23
10	{'C': 3, 'gamma': 1, 'kernel': 'rbf'}	0.153015	0.134113	0.151215	0.144144	0.145045	0.145507	0.006646	24
5	{'C': 1, 'gamma': 1, 'kernel': 'rbf'}	0.153915	0.133213	0.151215	0.145946	0.143243	0.145507	0.007207	24
20	{'C': 7, 'gamma': 1, 'kernel': 'rbf'}	0.153915	0.134113	0.153915	0.142342	0.141441	0.145146	0.007709	26
3	{'C': 0.1, 'gamma': 0.001, 'kernel': 'rbf'}	0.164716	0.144914	0.133213	0.113514	0.169369	0.145145	0.020556	27
25	{'C': 10, 'gamma': 1, 'kernel': 'rbf'}	0.153915	0.135014	0.153915	0.141441	0.140541	0.144965	0.007632	28
1	{'C': 0.1, 'gamma': 0.1, 'kernel': 'rbf'}	0.143114	0.133213	0.134113	0.136036	0.148649	0.139025	0.005936	29
0	{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}	0.129613	0.139514	0.129613	0.132432	0.133333	0.132901	0.003627	30