CSC411H Fall 2015 Facial Expression Prediction

Team Name: 😃😧😰😔Facial Expression Guessing Machine 🌝🌗🌚🌚

Name:	Ziyang Jiang	Zicong Zhou
Student Number:	1000085329	998487604
Email:	ziyang.jiang@mail.utoronto.ca	zicong.zhou@mail.utoronto.ca
Kaggle Name:	ziyang	Zicong
CDF Account:	c3jiangz	g3zhouzi

Introduction

The objective is to develop a model that predicts the facial expressions of the hidden test dataset, where there are 7 possible classifications. Our goal is to construct a model achieving persistent accuracy of at least 75% on the public test dataset. We have compared kNN, Logistic Regression, and the cluster algorithm based on SVM, Support Vector Classification(SVC). We have decided to use the latter, along with data preprocessing. The hyperparameters are chosen with cross validation.

Image Preprocessing

Standardization

The standardization function is implemented by the sklearn.preprocessing library. The SVC model assumes that the datasets are Gaussian distributed, and hence would perform poorly if given differently distributed data. Standardization scales the dataset into standard normalized data, in other words, Gaussian with zero mean and unit variance [1]. Another advantage is to avoid attributes in greater numeric ranges to dominate over smaller attributes [2].

C\ γ	0.001	0.005	0.01	0.05	0.1	0.5
1	[0.31128848346636262, 0.28471411901983662, 0.31356898517673887, 0.31687715269804823, 0.31958762886597936, 0.29304446978335236]					
2	[0.28456221198156684, 0.29506314580941445, 0.30804597701149428, 0.29540229885057473, 0.30102622576966931, 0.29304446978335236]					
3	[0.30663615560640733, 0.29532497149372861, 0.29405034324942791, 0.28604118993135014, 0.30558722919042192, 0.29942857142857143]					
4	[0.31018518518518517, 0.3276255707762557, 0.28538812785388129, 0.31314285714285717, 0.3002283105022831, 0.28734321550741165]					
5	[0.30102622576966931, 0.30330672748004561, 0.32023121387283238, 0.29701834862385323, 0.29907621247113164, 0.26865671641791045]					
6	[0.29128440366972475, 0.31506849315068491, 0.30892448512585813, 0.29714285714285715, 0.27968036529680368, 0.27137970353477764]					
7	[0.3230593607305936, 0.29680365296803651, 0.31771428571428573, 0.30355097365406641, 0.31729667812142037, 0.31536697247706424]					
8	[0.32151029748283755, 0.30821917808219179, 0.30628571428571427, 0.31885714285714284, 0.29874572405929306, 0.3047945205479452]					
9	[0.29828571428571427, 0.31314285714285717, 0.32611174458380843, 0.30160550458715596, 0.31314285714285717, 0.29344073647871116]					
10	[0.3127147766323024, 0.30549199084668194, 0.31221198156682028, 0.30549199084668194, 0.28260869565217389, 0.29646522234891676]					

C\ γ	0.001	0.005	0.01	0.05	0.1	0.5

1	[0.74800456100342072, 0.59703196347031962, 0.31500572737686139, 0.29919447640966629, 0.30102622576966931, 0.28865979381443296]
2	[0.75657142857142856, 0.63930635838150285, 0.33065442020665903, 0.31698973774230332, 0.31963470319634701, 0.28947368421052633]
3	[0.75598631698973773, 0.66705202312138734, 0.33640552995391704, 0.31356898517673887, 0.29418472063854045, 0.29142857142857143]
4	[0.76282782212086664, 0.65182648401826482, 0.32258064516129031, 0.31566820276497698, 0.29663962920046349, 0.29908675799086759]
5	[0.77319587628865982, 0.63521288837744538, 0.3249138920780712, 0.3093607305936073, 0.29760547320410491, 0.30434782608695654]
6	[0.74344355758266822, 0.6782407407407407, 0.34703196347031962, 0.29076396807297605, 0.27829099307159355, 0.30672748004561001]
7	[0.75712656784492594, 0.64367816091954022, 0.32220943613348679, 0.29223744292237441, 0.2986270022883295, 0.30149597238204834]
8	[0.78196347031963476, 0.67283236994219653, 0.35738831615120276, 0.31042382588774342, 0.28424657534246578, 0.27168949771689499]
9	[0.76484018264840181, 0.64861751152073732, 0.33142857142857141, 0.31235697940503432, 0.29337899543378998, 0.30389908256880732]
10	[0.77853881278538817, 0.63854047890535914, 0.33067274800456098, 0.31085714285714283, 0.28980526918671251, 0.30389908256880732]

We have performed cross validation on the original dataset and the scaled dataset with the SVC model over penalty parameter(C) from 1 to 10 and kernel coefficient(γ) from 0.001 to 0.5. The resulting mean accuracy of the original dataset is about 0.3010, where the scaled dataset is about 0.4400. The cross validation has shown that the SVC model performs better with the scaled dataset.

Histogram Equalization

C\ γ	0.001	0.005	0.01	0.05	0.1	0.5
1	[0.60481099656357384, 0.71542857142857141, 0.71297359357060852, 0.6571100917431193, 0.3614595210946408, 0.27822120866590649]					
2	[0.68072976054732037, 0.71925754060324831, 0.70608495981630315, 0.70171428571428573, 0.38768529076396807, 0.30320366132723114]					
3	[0.65743944636678198, 0.69248554913294802, 0.74022988505747123, 0.7013729977116705, 0.41714285714285715, 0.30285714285714288]					
4	[0.70967741935483875, 0.75862068965517238, 0.76632302405498287, 0.67844925883694418, 0.42775229357798167, 0.31927023945267957]					
5	[0.71627906976744182, 0.68786127167630062, 0.76598173515981738, 0.70091324200913241, 0.44228571428571428, 0.29862700228832951]					
6	[0.70353477765108319, 0.7226027397260274, 0.72633979475484611, 0.67388825541619157, 0.37785388127853881, 0.28392246294184721]					
7	[0.69555302166476629, 0.72716763005780349, 0.74628571428571433, 0.72405929304446981, 0.43264840182648401, 0.30786773090079816]					
8	[0.69415807560137455, 0.76598173515981738, 0.75258918296892985, 0.68721461187214616, 0.40364880273660203, 0.27042577675489066]					
9	[0.69159953970080557, 0.75629290617848965, 0.76396807297605474, 0.68493150684931503, 0.43542857142857144, 0.30160550458715596]					
10	[0.70923603192702389, 0.7229190421892816, 0.74541284403669728, 0.72063854047890541, 0.41391106043329534, 0.30365296803652969]					

Histogram Equalization is imported from the skimage.exposure library. The method enhances the contrast in an image by spreading out the most frequent intensity values in it [3]. A low contrast image has pixels clustered at a small range of intensities, the lack of change in the values of the dataset could lead to a poor classification model. After the equalization, the mean accuracy is approximately 0.5926, which is better than scaling alone. It has indicated that enhancing the contrast, variation in the intensity of the pixels, leads to a more accurate SVC model.

Normalization

C\ γ	0.001	0.005	0.01	0.05	0.1	0.5
1	[0.61687571265678454, 0.7094907407407407, 0.70239452679589509, 0.69460390355912749, 0.39215686274509803, 0.29304446978335236]					
2	[0.65792474344355756, 0.71917808219178081, 0.74971428571428567, 0.71641791044776115, 0.38558352402745993, 0.31657142857142856]					
3	[0.69634703196347036, 0.76027397260273977, 0.75604142692750287, 0.70776255707762559, 0.40981735159817351, 0.32302405498281789]					
4	[0.69690011481056258, 0.7160775370581528, 0.74541284403669728, 0.68077803203661325, 0.41570438799076215, 0.28848346636259975]					

5	[0.68264840182648401, 0.6990846681922197, 0.76457142857142857, 0.66361556064073224, 0.4132420091324201, 0.29506314580941445]
6	[0.7194982896237172, 0.76369863013698636, 0.74799541809851089, 0.66975666280417145, 0.40366972477064222, 0.29061784897025172]
7	[0.70923603192702389, 0.77028571428571424, 0.76201372997711669, 0.65437788018433185, 0.38646788990825687, 0.30672748004561001]
8	[0.69213226909920178, 0.74133949191685911, 0.76624857468643104, 0.69275028768699654, 0.38434982738780205, 0.27771428571428569]
9	[0.69873997709049251, 0.7640966628308401, 0.74628571428571433, 0.65981735159817356, 0.42955326460481097, 0.30672748004561001]
10	[0.75057471264367814, 0.75684931506849318, 0.76282782212086664, 0.70776255707762559, 0.38654503990877992, 0.28424657534246578]

The normalization function is also imported from the sklearn.preprocessing library. It scales the dataset to have unit norm, which is the base of the Vector Space Model (VSM). VSM is often used in clustering algorithms, including the SVC model [4]. Similarly, we have performed cross validation with varying C and γ combinations, and the mean accuracy is approximately 0.5944, which shows improvement comparing with the previous result, though marginal.

Principal Component Analysis

One of the matrix decomposition technique we have tried is PCA, implemented from the sklearn.decomposition library. The imported library performs linear dimensionality reduction using singular value decomposition to project the dataset to a lower dimensional space [5]. To best construct the PCA model, the unlabeled data are used for training, and the resulting model is used to transform the labeled data. The number of principal components(n) to keep is a hyperparameter that has to be experimented. Appendix *Figure 1* shows the number of components we have experimented, and the mean accuracy(score) received in the SVC model from performing cross validation.

As the graph has indicated, the maximum score is 0.751598383372 where n = 99. We have then proceeded with n fixed at 99, and performed cross validation on the SVC model with varying penalty parameter. However, the mean accuracies of the cross validation (*Figure 3*) fluctuates from 70-75%, which does not satisfy our goal; to be persistently over 75%. The reason it performs poorly on the SVC could be that the PCA significantly reduces the dimensionality, and spatial information is lost. On the other hand, SVC itself is effective in high dimensional spaces, and may require more spatial information in the training set to produce a definite model. Therefore, we have decided to disregard the dimensionality reduction, and train the SVC model based on the preprocessed images.

Models Comparison

Logistic Regression(LR)

The Logistic regression model we have built was implemented with the sklearn.linear_model library. Though it's a linear model for classification, multiclass

classification can be achieved with the One-vs-all LR. *Figure 4* is the cross validation we have performed to test the penalty parameter, C. From the result, C=0.005 returns the highest mean accuracy. Thus, in the next step, *Figure 5*, C is held constant at 0.005, and cross validation is performed over the number of components in the PCA. However, the result is similar as in the SVC, which all the scores are lower than using the original dataset.

The highest score obtained by cross validation in the LR is about 0.713, which is less than the SVC model. This could be due to the fact that LR assumes there exists a single linear decision boundary, but the dataset may require a more complicated decision function to analyze. On the other hand, the SVC has no such restriction, where different kernel functions such as 'poly' and 'rbf' could be deployed to create non-linear decision boundary.

kNN

The K-nearest neighbor model that we have built is implemented from the sklearn.neighbors library. In order to find the optimal hyperparameter, we first tried a wide range of k-value. The result of accuracy on the labeled dataset is show in Figure 6. We found our model performs better when the k-value is lower than 15, so we narrowed down the k-value as we show in Figure 7. From the cross validation, the best k-value that we found best fit our model is 7, and the accuracy is 56.459% on public test image. We have also performed PCA on the dataset prior the cross validation. Likewise, the accuracy did not improved.

The results from the cross validation on kNN performs not as well as SVC. Unlike Logistic regression, kNN is not restricted by a single decision boundary, however, it is sensitive to outliers. Outliers could easily lead to misclassification in a kNN model. The other reason could be that the distance property in the image is not as intrinsic, which limits the performance of the kNN model.

Support Vector Classification(SVC)

SVC is a clustering algorithm that provides an improvement based on the Support Vector Machines (6). The SVC creates decision boundary focusing on the support vectors(boundary points). To obtain nonlinear boundary, SVC can map the data into a higher-dimensional feature space, and the dot-product is computed as kernel functions. To configure the kernel function and hyperparameters that best fit the model for the given dataset, we have yet again performed cross validation over kernel='linear', 'poly', 'rbf', C=[1,10], and γ =[0.001, 0.5].

kernels = linear

max: 0.717217787913 mean: 0.687950002047 var: 0.000204323642604

C\ γ	0.001	0.005	0.01	0.05	0.1	0.5
1	[0.68871151653363738, 0.69977168949771684, 0.704337899543379, 0.68301026225769668, 0.69919632606199766, 0.67853042479908154]					
2	[0.70353477765108319, 0.69178082191780821, 0.67388825541619157, 0.68757126567844928, 0.70239452679589509, 0.68922018348623848]					
3	[0.71151653363740019, 0.69783352337514248, 0.66476624857468647, 0.7142857142857143, 0.69221967963386732, 0.69644902634593353]					
4	[0.66476624857468647, 0.68985176738882559, 0.68822170900692836, 0.69565217391304346, 0.67976878612716762, 0.67808219178082196]					
5	[0.68072976054732037, 0.69336384439359267, 0.64055299539170507, 0.68301026225769668, 0.66248574686431017, 0.66552511415525117]					
6	[0.67085714285714282, 0.69748858447488582, 0.67318757192174916, 0.7013729977116705, 0.68122786304604488, 0.69328703703703709]					
7	[0.70809578107183579, 0.68187001140250858, 0.70011402508551879, 0.68342857142857139, 0.68150684931506844, 0.6952054794520548]					
8	[0.671264367816092, 0.68009205983889531, 0.68757126567844928, 0.69017341040462432, 0.69327251995438999, 0.67356321839080457]					
9	[0.70472895040369088, 0.70353477765108319, 0.66590389016018303, 0.67505720823798632, 0.67620137299771166, 0.69460390355912749]					
10	[0.69257142857142862, 0.69485714285714284, 0.7172177879133409, 0.69610091743119262, 0.68922018348623848, 0.70239452679589509]					

When the kernel type is linear, naturally, the resulting boundary will be linear. The mean score of the cross validation is about 0.7172, which is the lowest in the 3 kernel functions we have tested. This indicates that the dataset could not be well classified with linear boundary.

kernels = poly

max: 0.7582668187 mean: 0.707148630828 var: 0.00491122678572

C\ γ	0.001	0.005	0.01	0.05	0.1	0.5
1	[0.41049030786773089, 0.74742857142857144, 0.73515981735159819, 0.7172177879133409, 0.74230330672748002, 0.75028636884306987]					
2	[0.45298165137614677, 0.71958285052143689, 0.73059360730593603, 0.72068965517241379, 0.74856486796785304, 0.74686431014823262]					
3	[0.5194508009153318, 0.71835803876852911, 0.74245939675174011, 0.73112128146453093, 0.73204104903078682, 0.71215596330275233]					
4	[0.58877434135166096, 0.73195876288659789, 0.75114155251141557, 0.72018348623853212, 0.74572405929304442, 0.74025974025974028]					
5	[0.59838895281933258, 0.73081328751431844, 0.72945205479452058, 0.75604142692750287, 0.72602739726027399, 0.73090079817559861]					
6	[0.61670480549190 8, 0.75143843498273877, 0.70459770114942533, 0.73888255416191562, 0.74885844748858443, 0.74230330672748002]					
7	[0.61714285714285 71, 0.72862029646522231, 0.72997711670480547, 0.73774230330672752, 0.74657534246575341, 0.72706422018348627]					
8	[0.63386727688787181, 0.7381776239907728, 0.7226027397260274, 0.73660205245153931, 0.75826681870011403, 0.70242214532871972]					
9	[0.62442922374429222, 0.74116305587229192, 0.73432155074116301, 0.75826681870011403, 0.73195876288659789, 0.72146118721461183]					
10	[0.62814645308924488, 0.74543378995433784, 0.72957422324510934, 0.72862029646522231, 0.73967889908256879, 0.73660205245153931]					

kernels = radial basis function(rbf)

max: 0.764840182648 mean: 0.593143185672 var: 0.0308003553809

C\ γ	0.001	0.005	0.01	0.05	0.1	0.5
1	[0.63428571428571423, 0.72643678160919545, 0.72177879133409351, 0.63013698630136983, 0.35844748858447489, 0.31771428571428573]					
2	[0.64994298745724055, 0.73179190751445089, 0.7263279445727483, 0.69284064665127021, 0.38084378563283922, 0.32155074116305588]					
3	[0.6915137614678899, 0.69988545246277201, 0.72342857142857142, 0.66020524515393386, 0.38417431192660551, 0.30102622576966931]					
4	[0.67719298245614035, 0.74141876430205955, 0.76484018264840181, 0.7208237986270023, 0.37286202964652221, 0.30216647662485746]					
5	[0.7126567844925884, 0.73401826484018262, 0.7451205510907003, 0.66020524515393386, 0.41550925925925924, 0.30593607305936071]					
6	[0.70675830469644907, 0.75115207373271886, 0.7408675799086758, 0.71543778801843316, 0.3995433789954338, 0.30320366132723114]					
7	[0.73744292237442921, 0.72695852534562211, 0.74857142857142855, 0.70378874856486795, 0.41008018327605955, 0.30285714285714288]					
8	[0.71247113163972287, 0.75428571428571434, 0.74770642201834858, 0.67958950969213228, 0.41095890410958902, 0.29324169530355099]					
9	[0.71526980482204361, 0.75484606613454963, 0.74828375286041193, 0.68378995433789957, 0.39747995418098508, 0.28308400460299193]					
10	[0.71921749136939006, 0.74572405929304442, 0.74230330672748002, 0.68197474167623418, 0.44013683010262256, 0.32648401826484019]					

Poly and rbf on the other hand, results in nonlinear boundary, which the model could still perform well when the relation between the class labels and attributes is nonlinear. The cross validation scores has show that the rbf returns the highest score at C=4, γ =0.01. We

then performed another cross validation which C and γ are held constant at 4 and 0.01, and looped for 10 time.

```
[ 0.73858447  0.74230331  0.74141876  0.75774971  0.72228571  0.76852908  0.77267509  0.77080958  0.73142857  0.74628571]
```

max 0.772675086108 mean: 0.74920699995 var: 0.000273426072077

The validation score shows that the mean is at 0.7492, which satisfies our goal that the accuracy should be persistent at about 75%. Therefore, the hyperparameters we have decided to use to construct the final model is C=4 and γ =0.01.

Summary

The models we have compared are not beneficial from PCA dimensionality reduction. From our findings, the linear models did not perform as well as the non-linear models, in other words the dataset may not be linearly separable. Among the models we've tested, SCV with the rbf kernel type performs the best, with hyperparameters C=4 and γ =0.01. Moreover, the constructed model gives an accuracy of 0.75598 on the public tests, which is satisfying.

Instruction to run the code

- Python Version: Python 2.7
- Install scipy, sklearn, and skimage packages
- The final model consists two files svm.py and util.py
- Change the path to the labeled, unlabeled, public, and hidden data in util.py functions load_labeled_data(), load_public_test(), load_unlabeled_data(), and load_hidden_test() to local paths

Appendix

Figure 1: Mean accuracy of the SVC cross validation with respect to the number of components in PCA

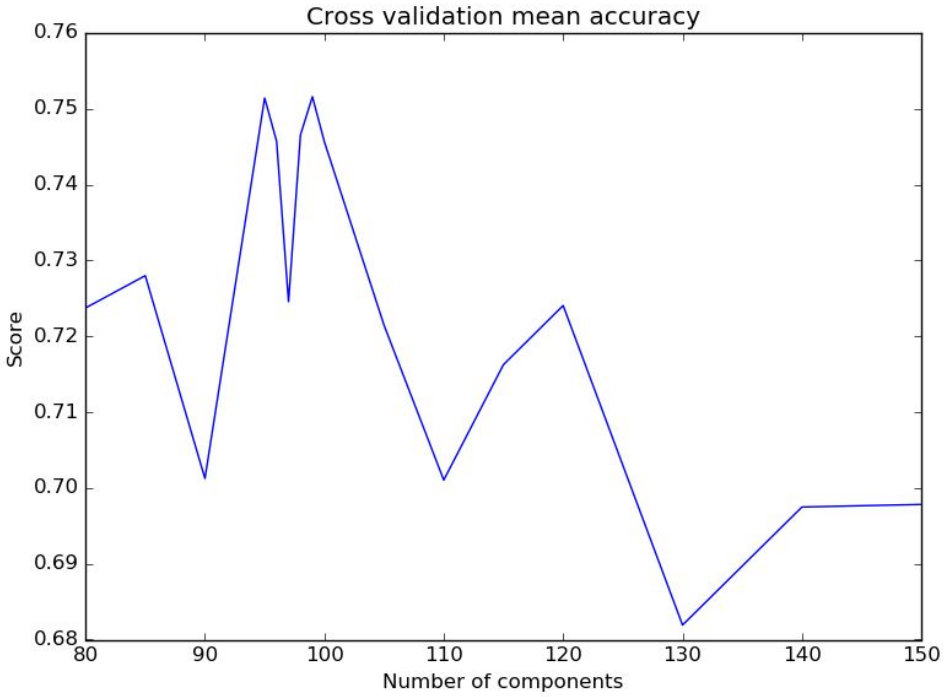


Figure 2: Eigenvectors where number of components = 99

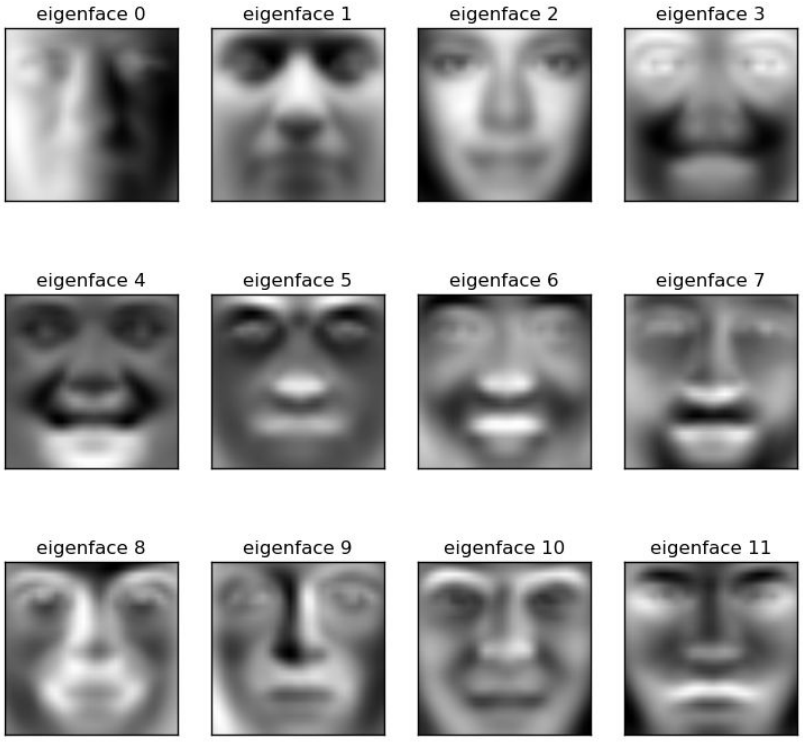


Figure 3. Mean accuracy of the SVC cross validation with respect to the penalty parameter where n = 99

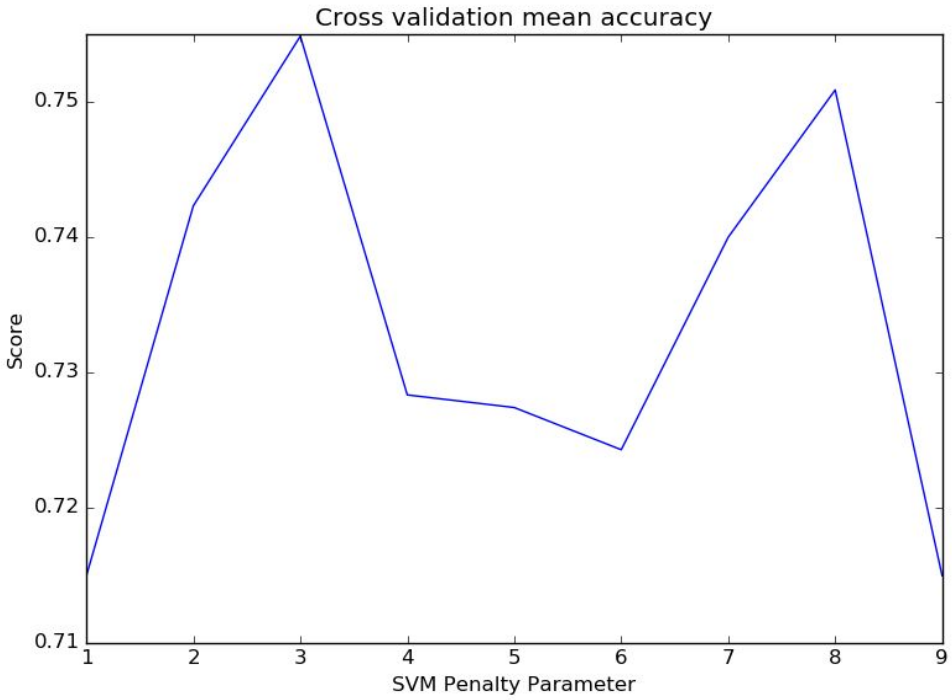


Figure 4. Mean accuracy of the Logistics Regression cross validation with respect to the penalty parameter

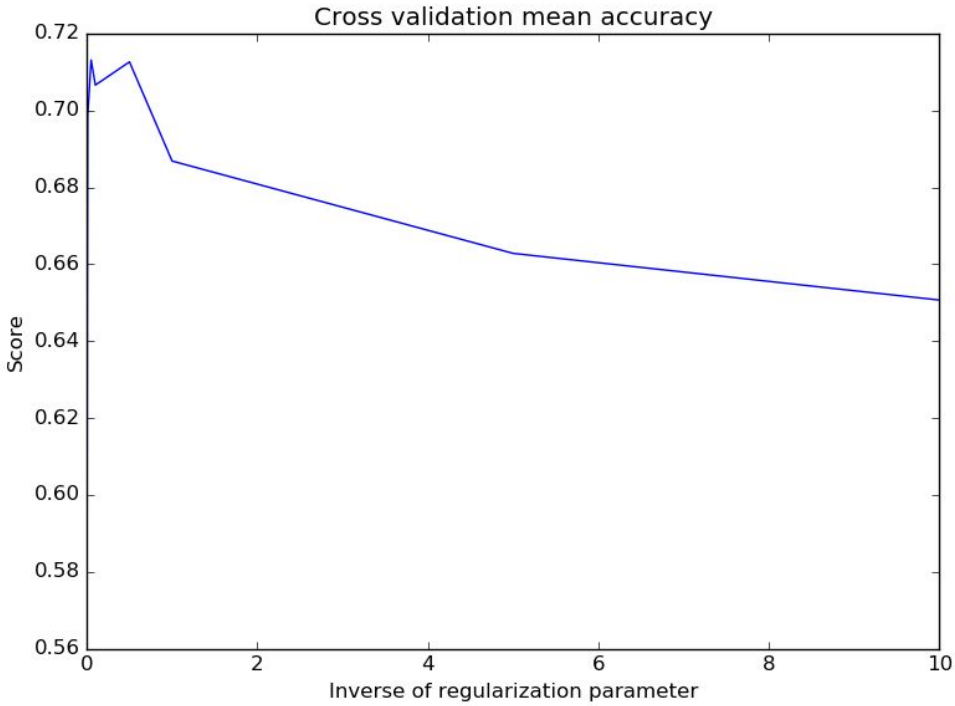


Figure 5. Mean accuracy of the Logistics Regression cross validation with respect to the number of components in PCA

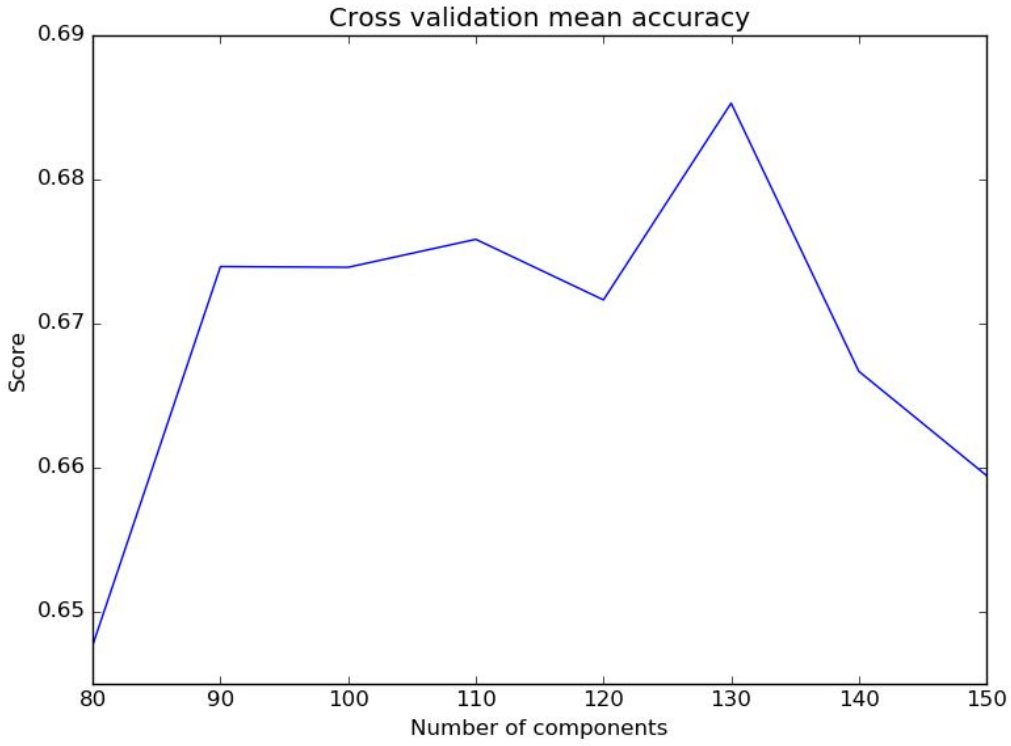


Figure 6. kNN model cross validation score with respect to the k-value

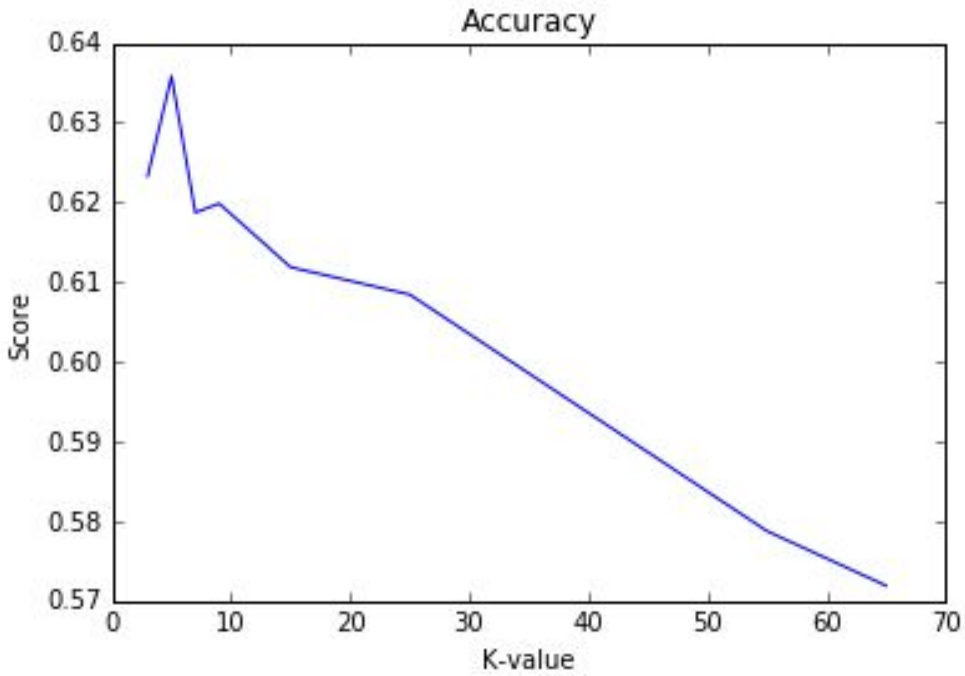
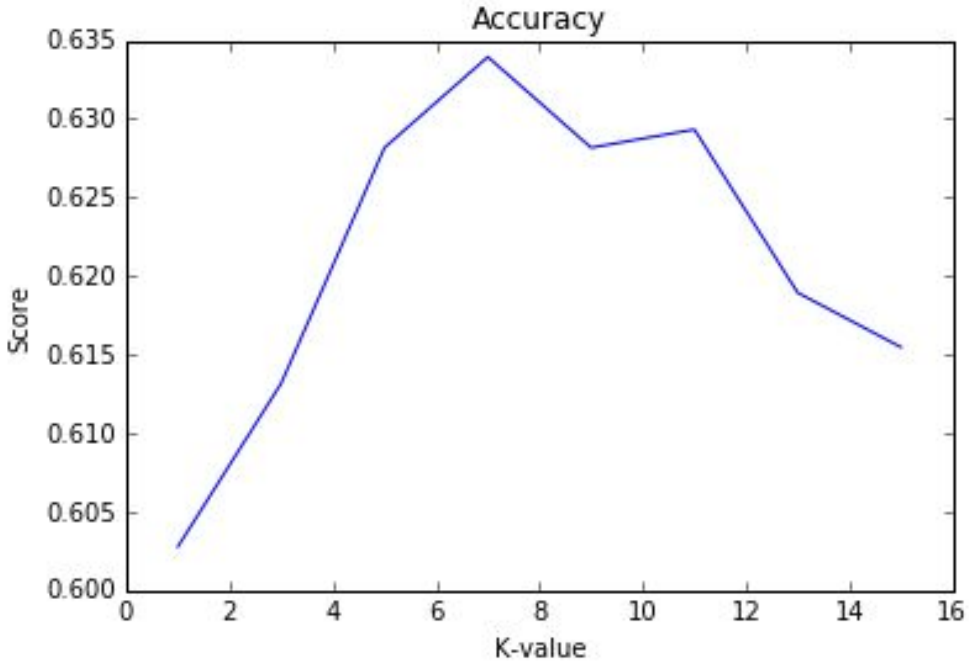


Figure 7. kNN model cross validation score with respect to the k-value



Bibliography

[1] Standardization: sklearn.preprocessing.scale
http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.scale.html

[2] A Practical Guide to Support Vector Classification
https://www.cs.sfu.ca/people/Faculty/teaching/726/spring11/svmguide.pdf

[3] Histogram Equalization
http://scikit-image.org/docs/dev/auto_examples/plot_equalize.html

[4] Normalization: sklearn.preprocessing.normalize
http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html#sklearn.preprocessing.normalize

[5] Principal Component Analysis: sklearn.decomposition.PCA
http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[6] Support Vector Machines: sklearn.svm.SVC
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html