|  |
| --- |
| BME Project Report |
| Image result for heart disease prediction |

|  |  |
| --- | --- |
| 8-5-2018 | Heart Disease Prediction System |

|  |
| --- |
| Heart diseases is hard to predict based on different symptoms, and there are 10-12 symptoms of a single heart disease. This makes it hard for the doctors to know weather a person has a disease or not. (c) jack\_1729 MIT License  SUBMITTED BY SHIVAM SHARMA (15UCS130) |

Project Report

Heart Disease Prediction System

# How to solve the problem using machine learning ?

The data is taken from :

<http://archive.ics.uci.edu/ml/datasets/heart+Disease>

As the data is less (303<1000) meaning (303 people) , thus I will use SVM to classify the data on the graph. If the data would have been more I would have used neural Net to solve the problem . Also the data of each patient has 13 attributes thus it has 13 dimensions to plot on a 2-d graph, thus I will use a technique called PCA to reduce the dimensions of the graph and bring it down to 2-d. My code does classification based on the values present in the data so as to predict the chances of heart disease of any patient.

## Process

It has 2 sets namely 0 for absence and 1 for presence where all the predicted values . The values between 1 and 4 are replaced to 1 to check the model performance.

1. Loading the dataset –

dataset = np.genfromtxt('processed.cleveland.data.txt',dtype = float, delimiter=',')

this line stores the dataset as a Numpy array.Taken from file – “'processed.cleveland.data.txt”

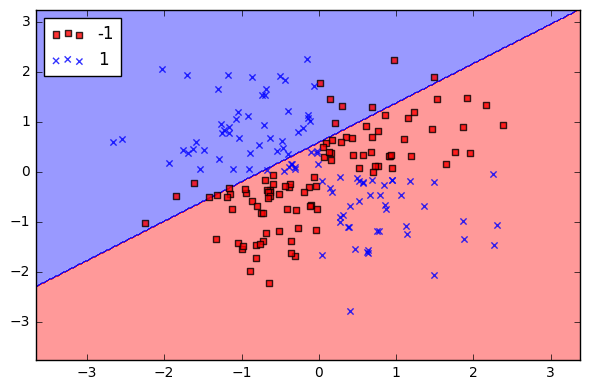
1. Taking out the feature Set and the Label Set –

X = dataset[:,0:12] #Feature Set

y = dataset[:,13] #Label Set

1. Creating a data model using LinearSVC –

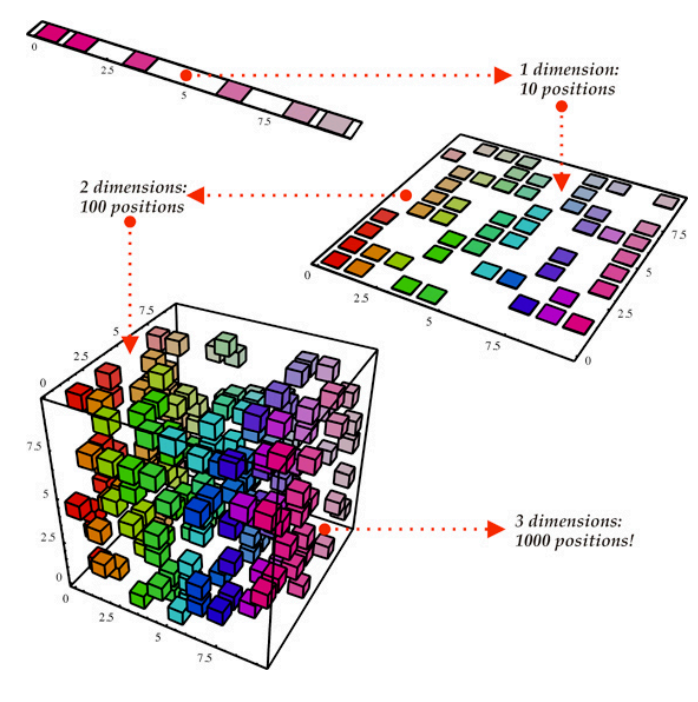
modelSVM = LinearSVC(C=0.1)



Classifying the data using a Linear SVM and predicting the probability of disease belonging to a particular class. SVC with parameter kernel=’linear’ & Penalty parameter C of the error term.

1. Putting dimension reduction and converting the data to 2-d –

pca = PCA(n\_components=2, whiten = True).fit(X)



When True (False by default) the components\_ vectors are multiplied by the square root of n\_samples and then divided by the singular values to ensure uncorrelated outputs with unit component-wise variances.

1. Applying cross validation on the training and test set for validating our Linear SVM Model-

X\_train,X\_test,y\_train,y\_test = cross\_validation.train\_test\_split(X\_new,y,test\_size=0.0,random\_state=0)

Cross Validation actually validates the data and trains it.

1. Training the model for a fixed number of epochs (iterations on a dataset).

modelSVM = modelSVM.fit(X\_train,y\_train)

Using the fit() function to train the data.

1. Testing the data on testing set –

skf = cross\_validation.StratifiedKFold(y,n\_folds=5)

Testing using cross validation with “Straightfied KFold” .

It divides the data into 5 parts , uses 4 for training the model and the remaining set to test the model.

TRAIN: [ 58 59 61 63 65 66 67 68 69 70 71 72 73 74 75 76 77 78

79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96

97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114

115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132

133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150

151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168

169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186

187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204

205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222

223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240

241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258

259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276

277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294

295 296 297 298 299 300 301 302]

TEST: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49

50 51 52 53 54 55 56 57 60 62 64]

TRAIN: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

54 55 56 57 60 62 64 122 123 124 125 126 127 128 129 130 131 132

133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150

151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168

169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186

187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204

205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222

223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240

241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258

259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276

277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294

295 296 297 298 299 300 301 302]

TEST: [ 58 59 61 63 65 66 67 68 69 70 71 72 73 74 75 76 77 78

79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96

97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114

115 116 117 118 119 120 121]

TRAIN: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71

72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107

108 109 110 111 112 113 114 115 116 117 118 119 120 121 179 182 183 185

186 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204

205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222

223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240

241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258

259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276

277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294

295 296 297 298 299 300 301 302]

TEST: [122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139

140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157

158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175

176 177 178 180 181 184 187]

TRAIN: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71

72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107

108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125

126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143

144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161

162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 180

181 184 187 239 240 241 242 244 249 250 251 252 253 254 255 256 257 258

259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276

277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294

295 296 297 298 299 300 301 302]

TEST: [179 182 183 185 186 188 189 190 191 192 193 194 195 196 197 198 199 200

201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218

219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236

237 238 243 245 246 247 248]

TRAIN: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71

72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107

108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125

126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143

144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161

162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179

180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197

198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215

216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233

234 235 236 237 238 243 245 246 247 248]

TEST: [239 240 241 242 244 249 250 251 252 253 254 255 256 257 258 259 260 261

262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279

280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297

298 299 300 301 302]

1. Making a plot–

xx,yy = np.meshgrid(np.arange(x\_min,x\_max,0.2),np.arange(y\_min,y\_max,0.2))

Makes plot with the minimum and maximum range.

1. Making counter –

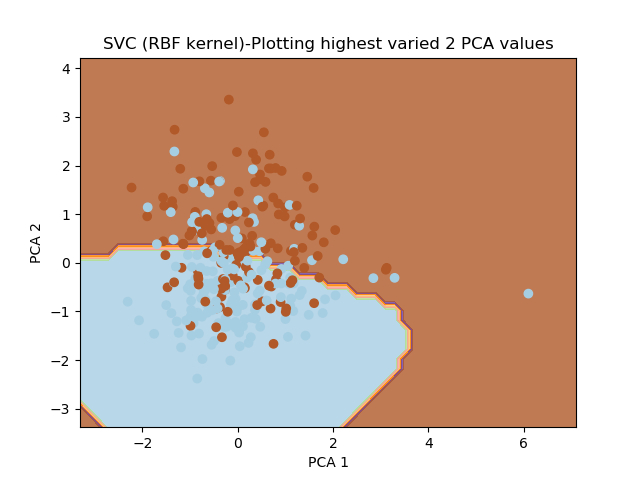
Z = Z.reshape(xx.shape) #reshaping Z using np array xx.

plt.contourf(xx,yy,Z,cmap = plt.cm.Paired,alpha=0.8) # making a counter with the parameters.

Makes a counter to the model. The plane that separates the diseased people in 13-d on a 2-d graph.

1. Showing the graph –

plt.show() # shows the graph.



The Light Blue region – Shows the people which have disease.

The Brown region – Shows the people which don’t have disease.

### Result

My model has made a counter and thus if we have the data of a new person we can test is easily using –

clf.score(X\_test,y\_test)

And plotting the result in a different color by using –

plt.scatter(X,Y1,color='red')

And thus I have completed my project which was really hard to do 10yrs back is now possible with Machine Learning by a University student like me.

Check the whole repository here –

<https://github.com/jack17529/Heart-Disease-Prediction>

References –

PCA -

1. An Introduction -<https://www.youtube.com/watch?v=jPmV3j1dAv4>

SVM -

1. In Detail - <https://www.youtube.com/watch?v=g8D5YL6cOSE>