Gabriel Buginga

Machine Learning
CPS 863

Lista 6

November 19, 2019

Date Performed:	November 19, 2019
Institution:	PESC/COPPE/UFRJ
Instructors:	Edmundo de Souza e Silva
	Rosa M. M. Leão
	Daniel Sadoc Menasché

The test set indices used were: [20, 27, 28, 38, 42, 50, 66, 67, 71, 79, 80, 85, 92, 96, 98, 99, 108, 112, 119, 120, 142, 157, 159, 169, 174, 194, 205, 208, 209, 227, 230, 232, 234, 235, 239, 248, 271, 276, 286, 288, 293, 300, 306, 313, 315, 318, 320, 324, 333, 356, 360, 365, 370, 371, 373, 375, 378, 393, 398, 405, 406, 409, 416, 420, 423, 461, 462, 468, 471, 472, 473, 490, 491, 493, 503, 512, 521, 525, 544, 546, 552, 561, 564, 566, 570, 574, 585, 587, 598, 602, 606, 608, 617, 623, 631, 636, 642, 659, 663, 670, 675, 680, 688, 694, 695, 696, 706, 709, 716, 721, 725, 728, 729, 734, 736, 741, 745, 754, 759, 764, 782, 786, 787, 789, 796, 797, 799, 808, 816, 831, 852, 853, 870, 873, 879, 891, 911, 918, 919, 928, 941, 942, 943, 955, 975, 983, 1005, 1024, 1039, 1042, 1053, 1054, 1055, 1056, 1057, 1080, 1093, 1098, 1103, 1110, 1116, 1120, 1136, 1146, 1149, 1153, 1154, 1160, 1164, 1166, 1168, 1171]. The training is just the rest 1000 points.

Problem 1

1: Figure 1 shows our results, using $\text{NLL}(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$ to get the log-likelihood and $\hat{\mathbf{w}}_{OLS} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$ to get the parameters, where we just increase our features when we increase the degree d. From the results we can see that the NLL and more importantly the predictive power measured with the mean-squared error on the test set stays very similar with degrees 1, 2 and 3; with 5, 10 and 15 the NLL increases (we don't gain better fitting) and the generalization starts to become worse showing decreased performance. This appointments make us conclude that d=1 is out best model, most simple that can example what the others can.

2: Now using two features, and noting that the "degrees" implies using interaction terms up to that degree, e.g. $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_1 x_2 + \cdots$. From figure 2 we see that the MSE on the test set hits a third of the previously value (27 to 8.67) already with degree 2, showing the importance of the interaction term. Greater model complexity don't add to much better generalization, so we keep Degree 2 model.

3:Finally, we use our 3 features to try to predict VO_2max. Results are shown at figure 3 (without any plots). The degree 2 model with all the features has test set MSE: 8.36, which don't add much

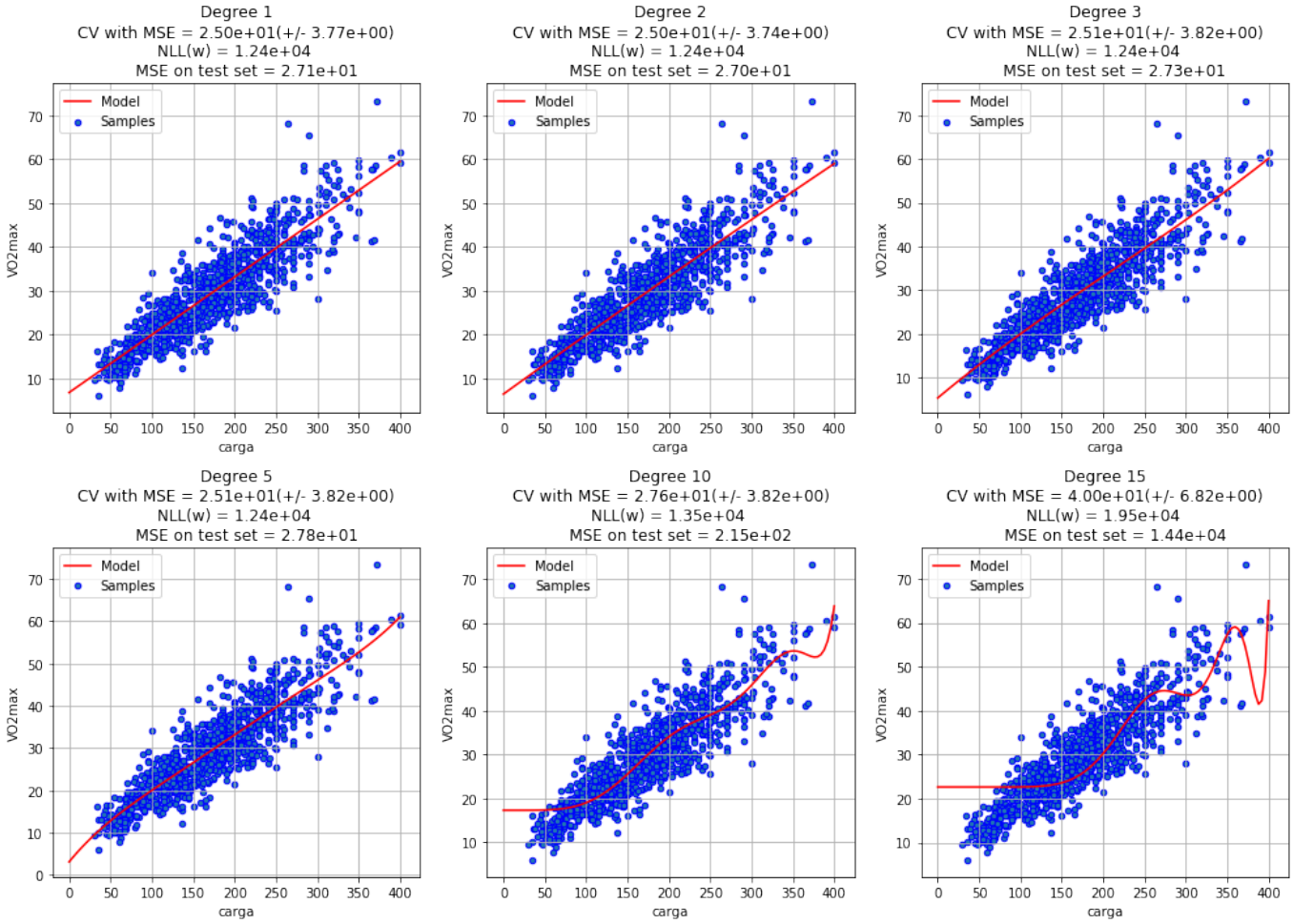

Figure 1: Problem 1's linear regression results with 'cargo' features.

to better generalization as the MSE from just two features as can be seen at figure 2 is 8.67 with 2 degrees; in other words the feature 'idade' don't carry much information about VO_2max that is not already with 'peso' and 'carga'.

4: Using $\text{VO}_2\,\text{max} \approx (W \times 11.4 + 260 + \text{peso} \times 3,5)/\text{peso}$ we obtain an MSE on the test set (exactly the same vector used for the above experiments) of 9.8969, greater than our best of 8.67 (degree 2 with 'peso' and 'carga' features). Our model with 'idade' only go to 8.36, still better; as already mentioned, 'idade' do not contain much information that is not already in 'carga' and 'peso'. This concludes, that our model was better overall. Exemplifying at the point ('carga','peso')=(202.0,67.1) we get $VO_2max = 41.6937$ with the provided equation, our model: 40.8563, true label: 41.430.

Problem 2

1:The MLE for the multivariate gaussian as [1] indicates is $\hat{\boldsymbol{\mu}}_{mle} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \triangleq \overline{\mathbf{x}}$ and $\hat{\boldsymbol{\Sigma}}_{mle} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T = \frac{1}{N}\left(\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T\right) - \overline{\mathbf{x}}\overline{\mathbf{x}}^T$. For the features 'carga' and VO_2max we end up with:

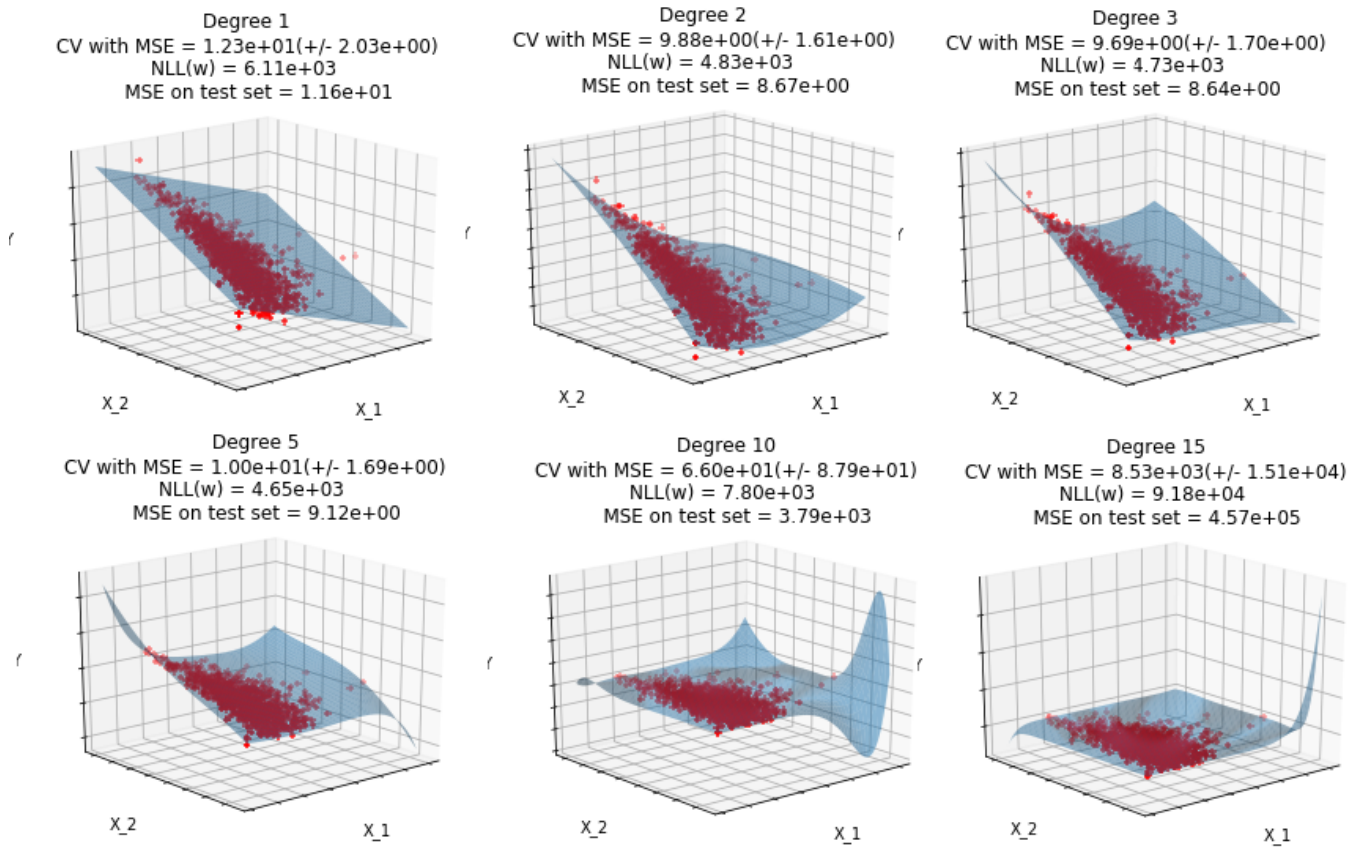

Figure 2: Problem 1's linear regression results with 'peso' and 'cargo' features.

```
Degree 1
CV with MSE = 1.18e+01(+/- 2.27e+00)
NLL(w) = 5.87e+03
    MSE on test set = 1.32e+01
Degree 2
CV with MSE = 9.40e+00(+/- 1.70e+00)
NLL(w) = 4.59e+03
    MSE on test set = 1.06e+01
Degree 3
CV with MSE = 9.58e+00(+/- 1.78e+00)
NLL(w) = 4.48e+03
    MSE on test set = 1.07e+01
Degree 5
CV with MSE = 1.61e+01(+/- 1.13e+01)
NLL(w) = 4.25e+03
    MSE on test set = 1.28e+01
Degree 10
CV with MSE = 3.75e+06(+/- 9.80e+06)
NLL(w) = 5.82e+03
    MSE on test set = 1.07e+02
Degree 15
CV with MSE = 3.37e+07(+/- 7.75e+07)
NLL(w) = 6.14e+04
    MSE on test set = 1.82e+03
```

Figure 3: Problem 1's linear regression results with 'peso','cargo' and 'idade' features.

$$\hat{\boldsymbol{\mu}}_{mle} = [172.0497 \quad 29.4152] \quad \hat{\boldsymbol{\Sigma}}_{mle} = \begin{bmatrix} 4960.9044 & 653.7960 \\ 653.7960 & 111.0169 \end{bmatrix}$$

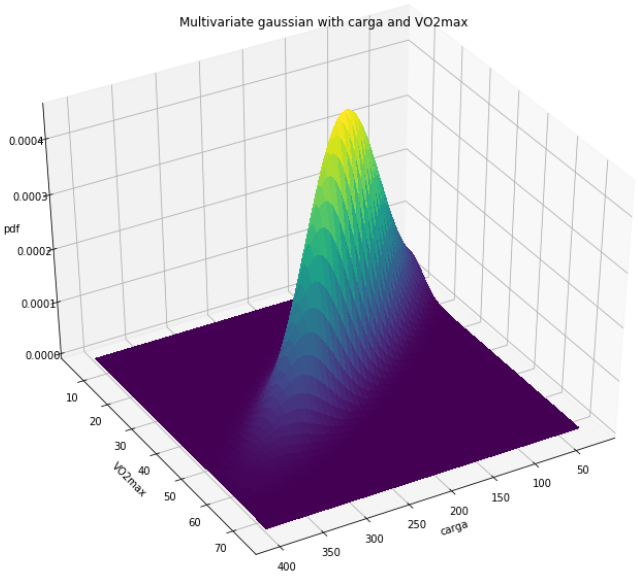

Figure 4: Problem 2's multivariate gaussian with 'carga' and 'VO2max'.

2:Using the features 'peso', 'carga' and 'VO_2max' we obtain the following MLE parameters:

$$\hat{\boldsymbol{\mu}}_{mle} = [85.6537 \quad 172.0497 \quad 29.4152] \quad \hat{\boldsymbol{\Sigma}}_{mle} = \begin{bmatrix} 211.2265 & 200.0376 & -24.2750 \\ 200.0376 & 4960.9044 & 653.7960 \\ -24.2750 & 653.7960 & 111.0169 \end{bmatrix}$$

To compare, we get the following points from test set for 'peso' and 'carga': $((67.1, 202.), (65.4, 95.), (63.1, 230.))$. Now we need to make a prediction for VO_2max based on our gaussian model, in order words, we need to make an inference: $\frac{p(x_1=67.1, x_2=202., x_3)}{p(x_1=67.1, x_2=202.)}$, with 'peso'=$x_1$, 'carga'=$x_2$ and $VO_2max = x_3$, then by symmetry of the gaussian, the expected value or the maximum probability is the same, making our prediction for x_3. Our true labels are [41.4307004521.1926605540.88748019], the regression model used was the degree 2 with features 'peso' and 'carga':

$$\begin{aligned} Gaussian \Rightarrow Ypred &= [38.28448276\ 23.53448276\ 43.28448276] \quad MSE = 7.0428 \\ Regression \Rightarrow Ypred &= [40.85635899\ 22.92485235\ 47.25460394] \quad MSE = 14.6235 \\ Equation \Rightarrow Ypred &= [41.69374069\ 24.0351682\ 49.17353407] \quad MSE = 25.60257 \end{aligned}$$

The Gaussian is the best of the three on those three points, followed by the regression, and then the equation. The equation made a huge mistake on the last point from 40.88 to 49.17, being an outlier in 'carga', it overshot; Regression too made a mistake, smaller nonetheless, by respecting the second order and possessing the interaction term it didn't suffer the outlier problem. The gaussian end up being the best because it has a natural propensity to be robust to outliers, it always brings data closer to its medium.

3:Similarly from the above:

$$\hat{\boldsymbol{\mu}}_{mle} = [53.239\ 85.65376\ 172.0497\ 29.41521379] \quad \hat{\boldsymbol{\Sigma}}_{mle} = \begin{bmatrix} 223.9318 & -32.1818 & -736.6330 & -100.4778 \\ -32.1818 & 211.2265 & 200.0376 & -24.2750 \\ -736.6330 & 200.0376 & 4960.9044 & 653.7960 \\ -100.4778 & -24.2750 & 653.7960 & 111.0169 \end{bmatrix}$$

Making the prediction using inference on the gaussian model, again as similar from **3**, we only get a slightly improvement on those 3 chosen test points with a MSE of 5.7911, with newly predicted points of [38.53448276 23.28448276 43.03448276]. Showing again how adding the 'idade' feature do not improve too much of the precision.

Problem 3

1: Diving the training data into the mentioned age brackets, we have the following models:

$$[18-40) \Rightarrow \hat{\boldsymbol{\mu}}_{mle} = [31.9700\ 85.9948\ 224.5175\ 37.1687] \quad \hat{\boldsymbol{\Sigma}}_{mle} = \begin{bmatrix} 30.3609 & 9.3130 & -3.7005 & -7.2927 \\ 9.3130 & 243.8167 & 54.2819 & -58.0854 \\ -3.7005 & 54.2819 & 3485.0185 & 444.7946 \\ -7.2927 & -58.0854 & 444.7946 & 93.6641 \end{bmatrix}$$

$$[40-60) \Rightarrow \hat{\boldsymbol{\mu}}_{mle} = [50.1320\ 88.1644\ 197.0553\ 32.3047] \quad \hat{\boldsymbol{\Sigma}}_{mle} = \begin{bmatrix} 35.1821 & -4.4294 & -152.7840 & -18.850 \\ -4.4294 & 209.6692 & 86.6734 & -45.207 \\ -152.7840 & 86.6734 & 3276.5810 & 426.279 \\ -18.8505 & -45.2071 & 426.2799 & 82.503 \end{bmatrix}$$

$$60 \geq \Rightarrow \hat{\boldsymbol{\mu}}_{mle} = [69.2238\ 82.2813\ 110.6586\ 21.3634] \quad \hat{\boldsymbol{\Sigma}}_{mle} = \begin{bmatrix} 48.2651 & -22.8984 & -178.1109 & -22.9834 \\ -22.8984 & 176.4987 & 129.6953 & -16.7231 \\ -178.1109 & 129.6953 & 1819.8526 & 245.3295 \\ -22.9834 & -16.7231 & 245.3295 & 47.8098 \end{bmatrix}$$

In order to predict VO_2max, we can just apply one of these models depending on the value of the feature 'idade'; in the case we don't have this feature we then need to apply Bayes rule together with the prior of each data point being inside each bracket.

2:Given a class, our Naive Bayes Classifier models each feature as being conditionally independent with a gaussian distribution, so:

$$p(\mathbf{x}|y=c,\boldsymbol{\theta}) = \prod_{j=1}^{D} \mathcal{N}\left(x_j|\mu_{jc},\sigma_{jc}^2\right)$$

$$p\left(\mathbf{x}_i,y_i|\boldsymbol{\theta}\right) = p\left(y_i|\boldsymbol{\pi}\right)\prod_j p\left(x_{ij}|\boldsymbol{\theta}_j\right) = \prod_c \pi_c^{\mathbb{I}(y_i=c)} \prod_j \prod_c p\left(x_{ij}|\boldsymbol{\theta}_{jc}\right)^{[(y_i=c)}$$

3:The specification is that the features are conditionally independent given the class label which is not what the gaussian model from Problem 2 postulates. The gaussian has in fact a interaction term of the features via the correlation matrix.

4:Writing the log-likelihood for a data point rightly gives us answers:

$$\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{c=1}^{C} N_c \log \pi_c + \sum_{j=1}^{D}\sum_{c=1}^{C}\sum_{i:y_i=c} \log p\left(x_{ij}|\boldsymbol{\theta}_{jc}\right)$$

Then $\hat{\pi}_c = \frac{N_c}{N}$, and the second term is just normal gaussian MLE but limited to each class data, in other words, μ_{jc} is the mean of feature j in objects of class c, and σ^2_{jc} its variance; all one-dimensional.

5:Our points came from the same training set used by all Problems above. $N = 1000$, $N_{id_1} = 200$, $N_{id_2} = 447$ and $N_{id_3} = 353$; noting that the features order is $j = 0$ as 'peso', $j = 1$ as 'carga' and $j = 2$ as VO_2max, then:

$$\begin{aligned}
Class\ id_1 \Rightarrow \hat{\pi}_{id_1} &= \frac{200}{1000} = 0.200 \\
\mu_{jid_1} &= [85.9948 \quad 224.5175 \quad 37.1687] \\
\sigma^2_{jid_1} &= [243.8167 \quad 3485.0185 \quad 93.6641] \\
Class\ id_2 \Rightarrow \hat{\pi}_{id_2} &= \frac{447}{1000} = 0.447 \\
\mu_{jid_2} &= [88.1644 \quad 197.0553 \quad 32.3047] \\
\sigma^2_{jid_2} &= [209.6692 \quad 3276.5810 \quad 82.5032] \\
Class\ id_3 \Rightarrow \hat{\pi}_{id_3} &= \frac{353}{1000} = 0.353 \\
\mu_{jid_3} &= [82.2813 \quad 110.6586 \quad 21.3634] \\
\sigma^2_{jid_3} &= [176.4987 \quad 1819.8526 \quad 47.8098]
\end{aligned}$$

Predict the class of the test set data points is just the posterior predictive for a class c: $p(y = c|\mathbf{x}, \mathcal{D}) \propto p(y = c|\mathcal{D}) \prod_{j=1}^{D} p\left(x_j|y = c, \mathcal{D}\right)$, where we can use our MLE estimates instead of using the whole bayesian procedure.

6:As mentioned, if we have a prior on each class, like $\hat{\pi}_c$, then is just a mixture of gaussians on each section of the data, hence we can do a posterior predictive too.

7:We can marginalize a data section creating a workable "class" (note that the gaussian is continuous, so we don't lose information), then is just bayesian manipulations to get there. So yes, we can too after some calculation and definitions.

Problem 4

1,2:The mixture of three gaussians has the following form; note that x_i now is ('peso','carga',VO_2max), and the data point got exchanged to have z_i indicating the cluster:

$$p\left(\mathbf{x}_i|\boldsymbol{\theta}\right) = \pi_{id_1}\mathcal{N}\left(\mathbf{x}_i|\boldsymbol{\mu}_{id_1}, \boldsymbol{\Sigma}_{id_1}\right) + \pi_{id_2}\mathcal{N}\left(\mathbf{x}_i|\boldsymbol{\mu}_{id_2}, \boldsymbol{\Sigma}_{id_2}\right) + \pi_{id_3}\mathcal{N}\left(\mathbf{x}_i|\boldsymbol{\mu}_{id_3}, \boldsymbol{\Sigma}_{id_3}\right)$$

All data for the mixture model is available, in other words, we don't need to apply the EM-algorithm. For the training the MLE end up being very similar to the Problem 3's Naive Bayes derivation, the only difference is that now the gaussians parameters are going to be with the full covariance matrices instead of being factored out:

$$
\begin{aligned}
Class\ id_1 \Rightarrow \hat{\pi}_{id_1} &= \frac{200}{1000} = 0.200 \\
\mu_{id_1} &= [85.9948 \quad 224.5175 \quad 37.1687] \\
\sigma^2_{id_1} &= \begin{bmatrix} 243.8167 & 54.2819 & -58.0854 \\ 54.2819 & 3485.0185 & 444.7946 \\ -58.0854 & 444.7946 & 93.6641 \end{bmatrix} \\
Class\ id_2 \Rightarrow \hat{\pi}_{id_2} &= \frac{447}{1000} = 0.447 \\
\mu_{id_2} &= [88.1644 \quad 197.0553 \quad 32.3047] \\
\sigma^2_{id_2} &= \begin{bmatrix} 209.6692 & 86.6734 & -45.2071 \\ 86.6734 & 3276.5810 & 426.2799 \\ -45.2071 & 426.2799 & 82.5032 \end{bmatrix} \\
Class\ id_3 \Rightarrow \hat{\pi}_{id_3} &= \frac{353}{1000} = 0.353 \\
\mu_{id_3} &= [82.2813 \quad 110.6586 \quad 21.3634] \\
\sigma^2_{id_3} &= \begin{bmatrix} 176.4987 & 129.6953 & -16.7231 \\ 129.6953 & 1819.8526 & 245.3295 \\ -16.7231 & 245.3295 & 47.8098 \end{bmatrix}
\end{aligned}
$$

3:As all data points, including the test points, have labelled clusters, to evaluate the probability of VO_2max we just need to check its cluster value z_i and then evaluate the pdf at that specific point. Note also that $p(x_3|x_1, x_2, z) = \frac{p(x_3,x_1,x_2,z)}{\sum_{x_3} p(x_3,x_1,x_2,z)} = \frac{p(x_3,x_1,x_2|z)p(z)}{\sum_{x_3} p(x_3,x_1,x_2|z)p(z)}$. Check Figure 4 for the results.

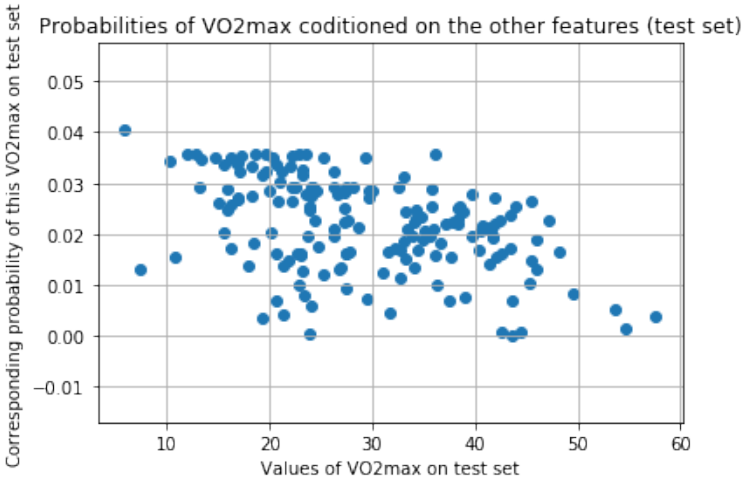

Figure 5: Problem 4's VO2max probabilities evaluation.

4: A good way to measure is with the posterior predictive on the cluster, i.e. if it classifies correctly the clusters on the test set when compared with Naive Bayes. The calculation of which cluster to choose is similar too, we need to compare the following equation for each class: $p(y = c|\mathbf{x}, \mathcal{D}) \propto p(y = c|\mathcal{D}) \prod_{j=1}^{D} p\left(x_j|y = c, \mathcal{D}\right)$, the class having greater probability is the one we assign. We compare just

counting the number of classified mistakes: Naive Bayes reaches on the test set error of 53%, and the mixture of gaussians reaches 39%, clearly better. Details with calculation check the code section.

5:By **4** the choice is for the mixture of gaussians. We are going to generate VO_2max by getting the mean of the posterior $p(x_3|x_1, x_2, z)$, which is also where its maximum probability value obtains, getting the first 10 test points:

$$\begin{aligned} TrueVO_2max =& [41.43070045 \quad 21.19266055 \quad 40.88748019 \quad 49.62686567 \quad 35.80060423 \\ & 37.08791209 \quad 45.4676259 \quad 40.47919294 \quad 32.68792711 \quad 22.91196388] \\ GeneratedVO_2max =& [38.78448276 \quad 22.53448276 \quad 43.53448276 \quad 43.28448276 \quad 34.03448276 \\ & 38.03448276 \quad 44.03448276 \quad 37.53448276 \quad 36.03448276 \quad 28.03448276] \\ MSE =& 10.8215 \end{aligned}$$

Problem 5

1,2:As we don't have the clusters labels we must use the EM-algorithm for a mixture of gaussians (noting that we can't use the 'idade' feature):

$$\begin{aligned} &\mathbf{E} - step \\ &\qquad r_{ik} = \frac{\pi_k p\left(\mathbf{x}_i|\boldsymbol{\theta}_k^{(t-1)}\right)}{\sum_{k'} \pi_{k'} p\left(\mathbf{x}_i|\boldsymbol{\theta}_{k'}^{(t-1)}\right)} \\ &\mathbf{M} - step \\ &\qquad \pi_k = \frac{1}{N}\sum_i r_{ik} = \frac{r_k}{N} \\ &\qquad \boldsymbol{\mu}_k = \frac{\sum_i r_{ik}\mathbf{x}_i}{r_k} \\ &\qquad \boldsymbol{\Sigma}_k = \frac{\sum_i r_{ik}\left(\mathbf{x}_i - \boldsymbol{\mu}_k\right)\left(\mathbf{x}_i - \boldsymbol{\mu}_k\right)^T}{r_k} = \frac{\sum_i r_{ik}\mathbf{x}_i\mathbf{x}_i^T}{r_k} - \boldsymbol{\mu}_k\boldsymbol{\mu}_k^T \end{aligned}$$

More than that, we don't know how many clusters is optimal, for that we use the information-theoretic criteria BIC. The results are plotted at figure 6. There we see indicated that the BIC chose 3 components with full covariance.

To label our point we can readily use a soft clustering approach, where r_{ik} is the responsibility of cluster k for point i. Then we can assign using hard clustering $z_i^* = \arg\max_k r_{ik}$:

$$r_{ik} \triangleq p\left(z_i = k|\mathbf{x}_i, \boldsymbol{\theta}\right) = \frac{p\left(z_i = k|\boldsymbol{\theta}\right) p\left(\mathbf{x}_i|z_i = k, \boldsymbol{\theta}\right)}{\sum_{k'=1}^{K} p\left(z_i = k'|\boldsymbol{\theta}\right) p\left(\mathbf{x}_i|z_i = k', \boldsymbol{\theta}\right)}$$

To see if our 3 clusters gather some meaning, we can check the distribution of the features that resides on each cluster. For that, we inspect visually inside figure 7 how the samples end up falling (hard clustering). There we see that the clusters for 'idade', the first two are similar, the third, older people, appear to have a statistical difference; maybe two similar middle-aged groups being fit and non-fit with the third showing systematic lower 'carga' and lower VO_2max which are the older people that has

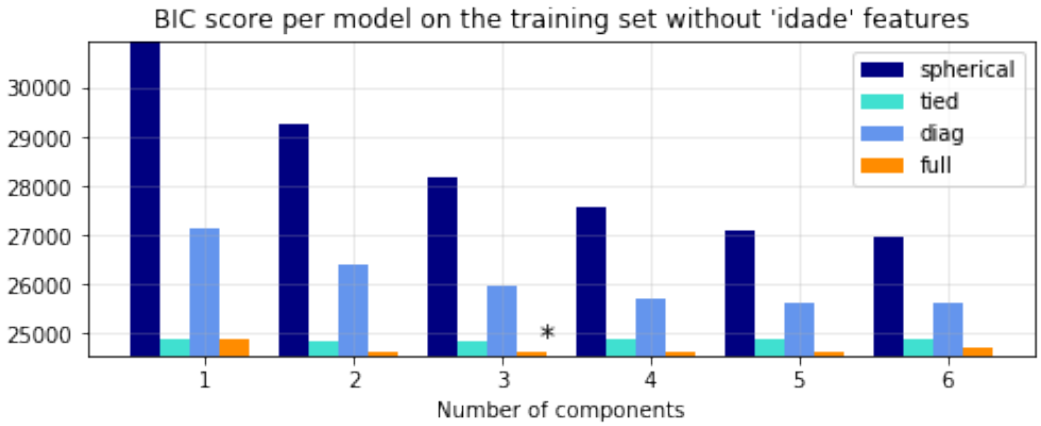

Figure 6: Problem 5's BIC results for the number of clusters, here 'components'.

bad performance anyway. Inspecting the 'peso' feature our suspicions gather more evidence, cluster 1 is the non-fit middle-aged group with corresponding low VO_2max and more 'peso', similar to the cluster 0 of the older people in its low VO_2max. Feature 'carga' correlates a lot with the VO_2max, again influencing the hypothesis that cluster 0 and 2 are both non-fit. Meanings are:

- cluster 2: middle-aged fit people (slim)
- cluster 1: middle-aged non-fit people (fat)
- cluster 0: older naturally non-fit people

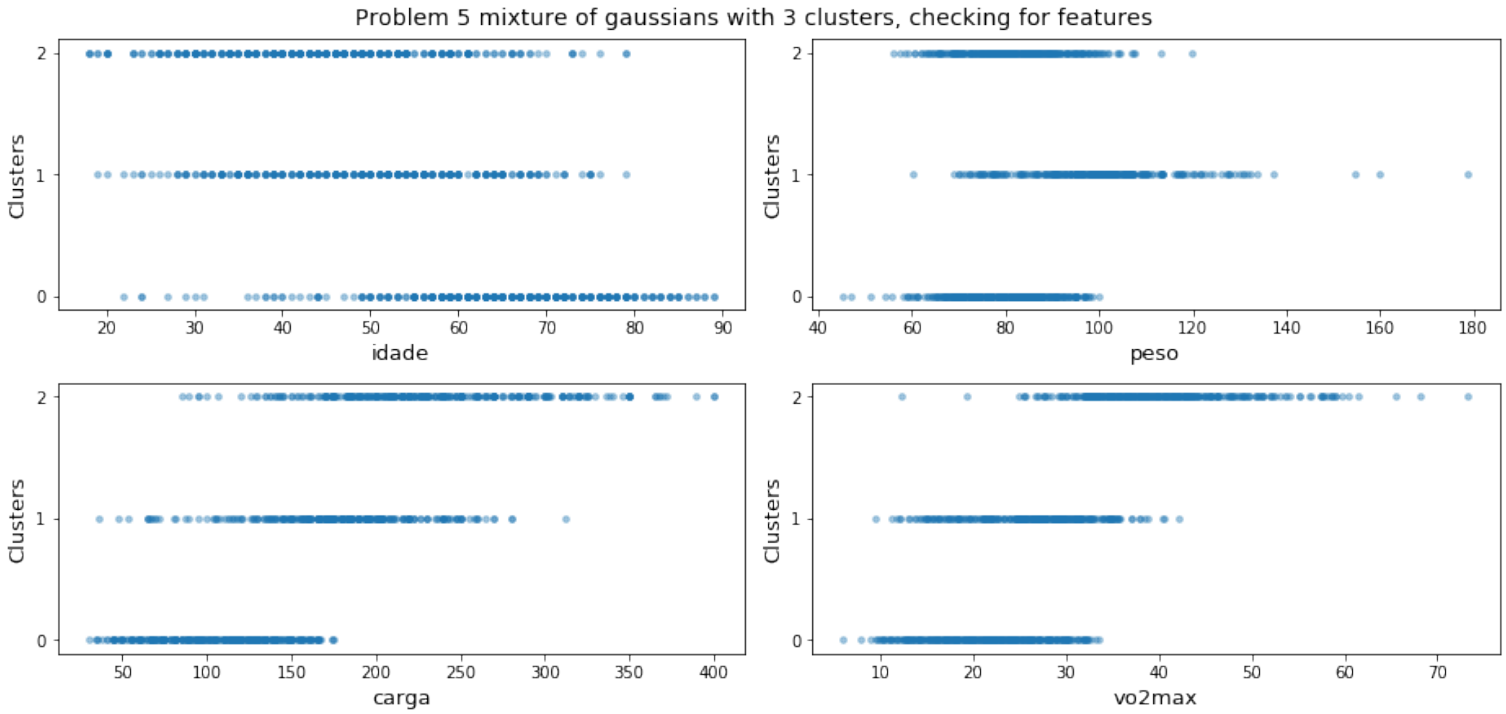

Figure 7: Problem 5's mixture of gaussians using 3 clusters, trained with EM algorithm.

Problem 6

1,2:We're going to use the mixture of gaussians with 3 clusters of Problem 5 (now with full data), besides being meaningful and interpretable as explained above on the last part of Problem's 5 solution, it is easier to do inference for any variable as we like, even generating data if the desire arrives. The parameters found:

$$
\begin{aligned}
Cluster\ 1 \Rightarrow \hat{\pi}_1 &= 0.3294 \\
\mu_1 &= [44.6494 \quad 93.0253 \quad 186.8353 \quad 29.1709] \\
\sigma_1^2 &= \begin{bmatrix} 151.3385 & 42.8937 & -186.7810 & -38.1369 \\ 42.8937 & 310.8545 & 91.6045 & -43.4442 \\ -186.7810 & 91.6045 & 1915.6070 & 226.8291 \\ -38.1369 & -43.4442 & 226.8291 & 45.6181 \end{bmatrix} \\
Cluster\ 2 \Rightarrow \hat{\pi}_2 &= 0.2634 \\
\mu_2 &= [44.5175 \quad 82.7969 \quad 242.8636 \quad 41.0185] \\
\sigma_2^2 &= \begin{bmatrix} 106.4581 & -1.0555 & -232.6167 & -35.2810 \\ -1.0555 & 113.3932 & 144.2008 & -38.2284 \\ -232.6167 & 144.2008 & 3467.9754 & 368.2551 \\ -35.2810 & -38.2284 & 368.2551 & 78.0742 \end{bmatrix} \\
Cluster\ 3 \Rightarrow \hat{\pi}_3 &= 0.4072 \\
\mu_3 &= [65.8347 \quad 81.5362 \quad 114.2599 \quad 22.1046] \\
\sigma_3^2 &= \begin{bmatrix} 90.5646 & -26.0193 & -275.6613 & -37.0079 \\ -26.0193 & 127.1634 & 128.1836 & -6.8650 \\ -275.6613 & 128.1836 & 1618.0282 & 231.2789 \\ -37.0079 & -6.8650 & 231.2789 & 44.3771 \end{bmatrix}
\end{aligned}
$$

2:With ('idade','peso','carga','VO_2max')=(x_1, x_2, x_3, x_4). The requirement is that we're able to calculate $p(x_4|40 \leqslant x_1 < 50)$; with our model we can calculate $p(x_1, x_2, x_3, x_4)$, then it's direct to obtain it via bayes and inference:

$$
p(x_4|40 \leqslant x_1 < 50) = \frac{\int p(40 \leqslant x_1 < 50, x_2, x_3, x_4) dx_2 dx_3}{\int p(40 \leqslant x_1 < 50, x_2, x_3, x_4) dx_2 dx_3 dx_4} \tag{1}
$$

3:We want an inference for (x_2, x_3, x_4)=(82.5, 181, 32.6).

$$
\begin{aligned}
p(40 \leqslant x_1 < 50|x_2 = 82.5, x_3 = 181, x_4 = 32.6) &= \frac{p(40 \leqslant x_1 < 50, x_2 = 82.5, x_3 = 181, x_4 = 32.6)}{\int p(x_1, x_2 = 82.5, x_3 = 181, x_4 = 32.6) dx_1} \\
&= 0.312797 \\
p(50 < x_1 < 60|x_2 = 82.5, x_3 = 181, x_4 = 32.6) &= 0.351502 \\
p(60 \leqslant x_1 \leqslant 70|x_2 = 82.5, x_3 = 181, x_4 = 32.6) &= 0.113142
\end{aligned}
$$

Problem 7

1,2: The K-means algorithm with 3 and 4 clusters are shown at figures 8 and 9.

3,4: The clusters clearly have information about 'idade' brackets. Figure 8 with 3 clusters shows cluster 1 (with a nice peak at "[30-50)") and 2 located at younger ages (maybe they are the younger

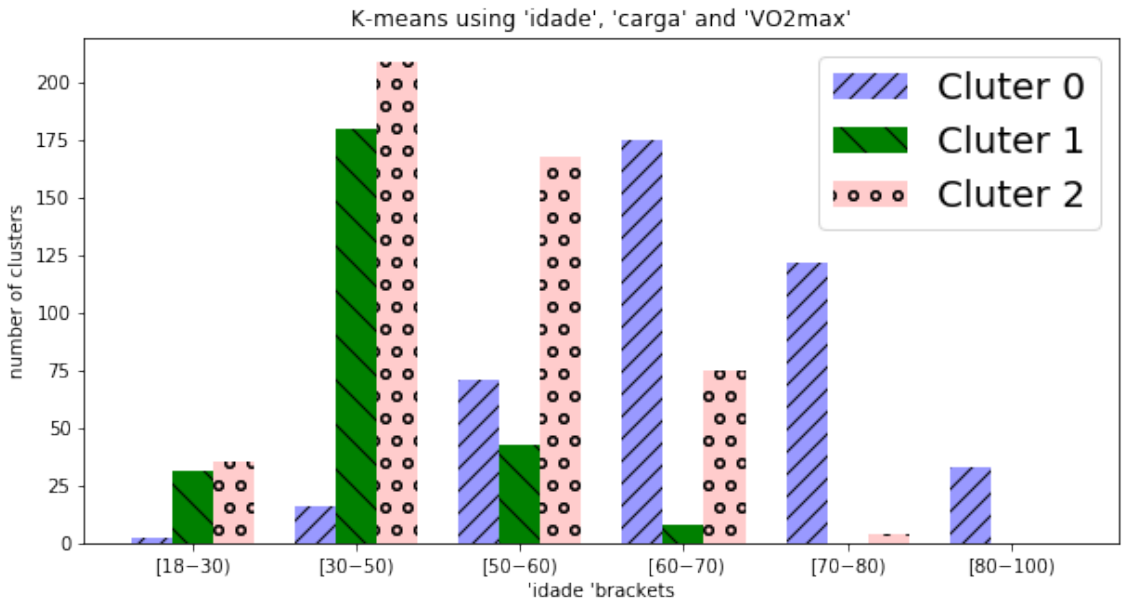

Figure 8: Problem 7's k-means with 3 clusters.

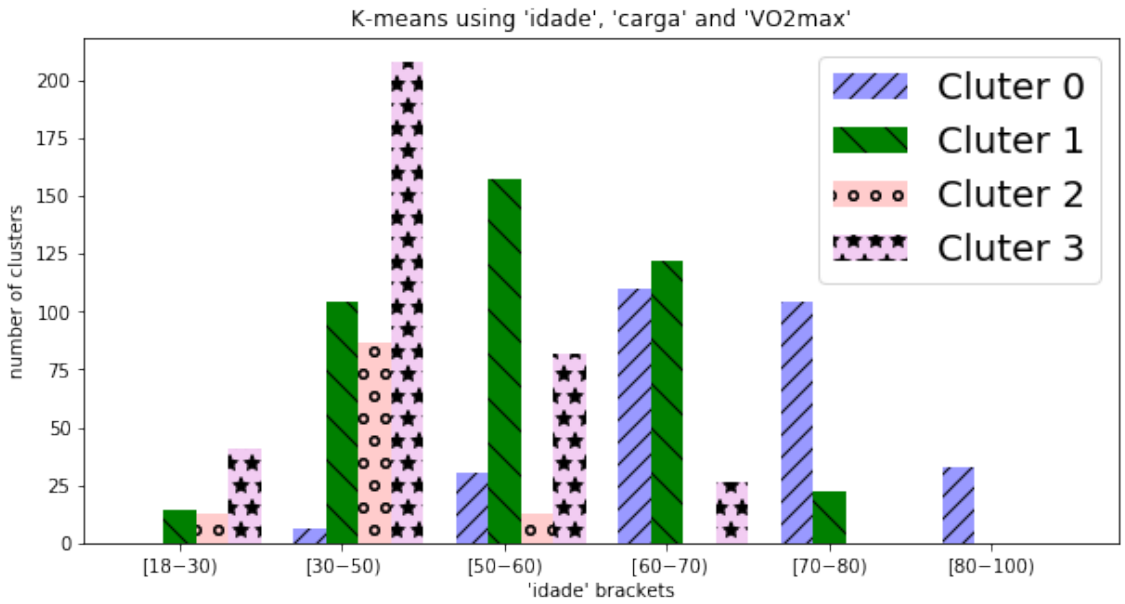

Figure 9: Problem 7's k-means with 4 clusters.

fit and non-fit), cluster 0 has concentration of older ages (the older ages with non-fit physical situation overall). For figure 9, cluster 0 still has that same meaning, cluster 2 peaked at "[30-50)"; cluster 1 and 3 appear to be a more fine-grained version of the above cluster 2. Concluding, k-means really got some explanatory information from our data, showing its usefulness; further detailed analysis can better show the semantics.

Code in Python

All codes can be run in the order of the problems.

Problem 1

```
from matplotlib import pyplot as plt
%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
from IPython.core.pylabtools import figsize
from sklearn.calibration import calibration_curve
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.datasets import load_boston
from sklearn import datasets
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.metrics import log_loss
import numpy as np
import pandas as pd
import os
current_path = os.getcwd()
exp_file_2 = '\\datasets\\Dados-medicos.txt'
df = pd.DataFrame()
df = pd.read_csv(current_path + exp_file_2 ,comment =\
 '#',header=None,delimiter='␣')
data_ = df.values

data = np.zeros([data_.shape[0],4])
for i,v in enumerate(data_):
    data[i] = data_[i][[0,3,4,5]]

choice = np.random.choice(data.shape[0],data.shape[0]-172,replace=False)
t = np.zeros(data.shape[0])
t[choice]=1

train = data[choice]
ind = []
for i,v in enumerate(data):
    if i not in choice:
        ind.append(i)
test = data[ind]

X = train[:,2].copy()
y = train[:,3].copy()
X_test = test[:,2].copy()
y_test = test[:,3].copy()

plt.figure(figsize=(14, 10))
degrees = [1, 2, 3, 5, 10, 15]
for i in range(len(degrees)):
    ax = plt.subplot(2, 3, i + 1)
```

```
    #plt.setp(ax, xticks=(), yticks=())

    polynomial_features = PolynomialFeatures(degree=degrees[i],
                                             include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])
    pipeline.fit(X[:, np.newaxis], y)

    # Evaluate the models using crossvalidation
    scores = cross_val_score(pipeline, X[:, np.newaxis], y,
                             scoring="neg_mean_squared_error", cv=10)


    y_pred = pipeline.predict(X[:, np.newaxis])
    NLL = (((y - y_pred)**2).sum())/2

    y_pred = pipeline.predict(X_test[:, np.newaxis])
    MSE_test = (((y_test - y_pred)**2).sum())/len(y_test)

    X_func = np.linspace(0, np.max(X[:]), 100)
    plt.plot(X_func, pipeline.predict(X_func[:, np.newaxis]),\
     label="Model",color='r')
    #plt.plot(X_func, true_fun(X_func), label="True function")
    plt.scatter(X, y, edgecolor='b', s=20, label="Samples")
    plt.xlabel("carga")
    plt.ylabel("VO2max")
    #plt.xlim((0, 1))
    #plt.ylim((-2, 2))
    plt.legend(loc="best")
    plt.grid(True)
    plt.title("Degree {}\nCV with MSE = {:.2e}(+/- {:.2e})\nNLL(w) = {:.2e} \n\
    MSE on test set = {:.2e}".format(
        degrees[i], -scores.mean(), scores.std(), NLL, MSE_test))
    #plt.autoscale(tight=True)
plt.tight_layout()
plt.show()

X = train[:,1:3].copy()
y = train[:,3].copy()
X_test = test[:,1:3].copy()
y_test = test[:,3].copy()

W = test[:,2].copy()
peso = test[:,1].copy()
y_pred = (W*11.4 + 260 + peso*3.5)/peso
MSE_test = (((y_test - y_pred)**2).sum())/len(y_test)

degrees = [1, 2, 3, 5,10,15]
X_train = X
y_train = y
for i in range(len(degrees)):
    #ax = plt.subplot(2, 3, i + 1,projection='3d')
```

```
    #plt.setp(ax, xticks=(), yticks=())

    polynomial_features = PolynomialFeatures(degree=degrees[i],
                                             include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])
    pipeline.fit(X, y)

    # Evaluate the models using crossvalidation
    scores = cross_val_score(pipeline, X, y,
                             scoring="neg_mean_squared_error", cv=10)


    y_pred = pipeline.predict(X)
    NLL = (((y - y_pred)**2).sum())/2

    y_pred = pipeline.predict(X_test)
    MSE_test = (((y_test - y_pred)**2).sum())/len(y_test)

    elev = 20
    azim = -130
    plot_figs(i+1, elev, azim, X, pipeline)

    plt.title("Degree {}\nCV with MSE = {:.2e}(+/- {:.2e})\nNLL(w) = {:.2e} \n\
    MSE on test set = {:.2e}".format(
        degrees[i], -scores.mean(), scores.std(), NLL, MSE_test))
    #X_func = np.linspace(0, np.max(X[:,1]), 100)
    #plt.plot(X_func, pipeline.predict(X_func[:, np.newaxis]), \
    ="Model",color='r')

#plt.tight_layout()
plt.show()

X = train[:,0:3].copy()
y = train[:,3].copy()
X_test = test[:,0:3].copy()
y_test = test[:,3].copy()

#All data
degrees = [1, 2, 3, 5,10,15]
X_train = X
y_train = y
for i in range(len(degrees)):

    polynomial_features = PolynomialFeatures(degree=degrees[i],
                                             include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])
    pipeline.fit(X, y)

    # Evaluate the models using crossvalidation
```

```
    scores = cross_val_score(pipeline, X, y,
                             scoring="neg_mean_squared_error", cv=10)

    y_pred = pipeline.predict(X)
    NLL = (((y - y_pred)**2).sum())/2

    y_pred = pipeline.predict(X_test)
    MSE_test = (((y_test - y_pred)**2).sum())/len(y_test)

    print("Degree {}\nCV with MSE = {:.2e}(+/- {:.2e})\nNLL(w) = {:.2e} \n\
    MSE on test set = {:.2e}".format(
        degrees[i], -scores.mean(), scores.std(), NLL, MSE_test))

W = test[0,2].copy()
peso = test[0,1].copy()
y_pred = (W*11.4 + 260 + peso*3.5)/peso
print(W,peso,y_pred,"True",test[0,3])

X = train[:,1:3].copy()
y = train[:,3].copy()
polynomial_features = PolynomialFeatures(degree=2,
                                         include_bias=False)
linear_regression = LinearRegression()
pipeline = Pipeline([("polynomial_features", polynomial_features),
                     ("linear_regression", linear_regression)])
pipeline.fit(X, y)
pipeline.predict([test[0,1:3]])
```

Problem 2

```
from scipy.stats import multivariate_normal
#subquestao 1
X = train[:,2:]
mean = np.mean(X, axis=0)
cov = np.cov(X, rowvar=0)
gd = multivariate_normal(mean=mean,cov = cov)

fig = plt.figure(1, figsize=(10, 8))
plt.clf()
elev = 40
azim = 60
ax = Axes3D(fig, rect=[0, 0, 1, 1],elev=elev, azim=azim)

x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5

X_plane = np.arange(x_min, x_max, 0.25)
Y_plane = np.arange(y_min, y_max, 0.25)
X_plane, Y_plane = np.meshgrid(X_plane, Y_plane)

c = np.ravel(X_plane).shape[0]
Z = np.zeros((c,2))
for i,v in enumerate(np.ravel(Y_plane)):
    Z[i]=[np.ravel(X_plane)[i],np.ravel(Y_plane)[i]]
```

```
#pos = np.empty(X_plane.shape + (2,))
#pos[:, :, 0] = X_plane; pos[:, :, 1] = Y_plane
#ax.plot_surface(X_plane, Y_plane, gd.pdf(pos),cmap='viridis')

ax.plot_surface(X_plane,
                Y_plane,
                (gd.pdf(Z)).reshape(X_plane.shape),
                cmap='viridis', linewidth=0, antialiased=False)
ax.set_xlabel('carga')
ax.set_ylabel('VO2max')
ax.set_zlabel('pdf')
plt.title("Multivariate gaussian with carga and VO2max")

#subquestao 2
X = train[:,1:3]
mean = np.mean(X, axis=0)
cov = np.cov(X, rowvar=0)
gd_2 = multivariate_normal(mean=mean,cov = cov)


X = train[:,1:].copy()
mean = np.mean(X, axis=0)
cov = np.cov(X, rowvar=0)
gd_3 = multivariate_normal(mean=mean,cov = cov)

x_true = test[0:3,1:3]
y_true = test[0:3,3]
print(x_true,y_true)

x_min, x_max = train[:, 3].min() - .5, train[:, 3].max() + .5
x = np.arange(x_min,x_max,0.25)
y_pred = np.zeros(y_true.shape)

for j,xi in enumerate(x_true):
    x3 = np.zeros([x.shape[0],3])
    for i,v in enumerate(x):
        x3[i]=[xi[0],xi[1],v]

    y = gd_3.pdf(x3)/gd_2.pdf([xi[0],xi[1]])
    plt.plot(x,y)
    print (x3[np.argmax(y)])
    y_pred[j] =  x3[np.argmax(y)][2]

print (y_pred,y_true)
print ("MSE_Gaussian",(((y_true - y_pred)**2).sum())/len(y_true))

#modelo preferido da Questao 1
X = train[:,1:3].copy()
y = train[:,3].copy()

polynomial_features = PolynomialFeatures(degree=2,
                                         include_bias=False)
```

```
linear_regression = LinearRegression()
pipeline = Pipeline([("polynomial_features", polynomial_features),
                     ("linear_regression", linear_regression)])
pipeline.fit(X, y)

y_pred = pipeline.predict(x_true)
print (y_pred)
print ("MSE_Regression",(((y_true - y_pred)**2).sum())/len(y_true))

W = x_true[:,1]
peso = x_true[:,0]
y_pred = (W*11.4 + 260 + peso*3.5)/peso
print (y_pred)
print ("MSE_Equation",(((y_true - y_pred)**2).sum())/len(y_true))

#subquestao 3
X = train[:,0:].copy()
mean = np.mean(X, axis=0)
cov = np.cov(X, rowvar=0)
gd_4 = multivariate_normal(mean=mean,cov = cov)

x_true = test[0:3,0:3]
y_true = test[0:3,3]
y_true
x_min, x_max = train[:, 3].min() - .5, train[:, 3].max() + .5
x = np.arange(x_min,x_max,0.25)
y_pred = np.zeros(y_true.shape)

for j,xi in enumerate(x_true):
    x3 = np.zeros([x.shape[0],4])
    for i,v in enumerate(x):
        x3[i]=[xi[0],xi[1],xi[2],v]

    y = gd_4.pdf(x3)/1
    plt.plot(x,y)
    print (x3[np.argmax(y)])
    y_pred[j] =  x3[np.argmax(y)][3]

print (y_pred,y_true)
print ("MSE_Gaussian",(((y_true - y_pred)**2).sum())/len(y_true))
```

Problem 3

```
#Questao 3
id1=[]
id2=[]
id3=[]

for i,v in enumerate(train):
    if v[0]>=18 and v[0]<40:
        id1.append(i)
    if v[0]>=40 and v[0]<60:
        id2.append(i)
    if v[0]>=60:
```

```
        id3.append(i)

train_id1=train[id1]
train_id2=train[id2]
train_id3=train[id3]

for i,X in enumerate([train_id1,train_id2,train_id3]):
    print ("Faixa id",i+1)
    mean = np.mean(X, axis=0)
    cov = np.cov(X, rowvar=0)
    gd = multivariate_normal(mean=mean,cov = cov)
    print ("[{:.4f} \; {:.4f} \; {:.4f} \; {:.4f}]".\
    format(mean[0],mean[1],mean[2],mean[3]))
    for i in cov:
        print ("{:.4f} & {:.4f} & {:.4f} & {:.4f} \\\\".\
        format(i[0],i[1],i[2],i[3]))

nc, pic = np.zeros([3]), np.zeros([3])
nc[0] = len(train_id1)
nc[1] = len(train_id2)
nc[2] = len(train_id3)
print (nc)
for i,_ in enumerate(nc):
    pic[i]=nc[i]/len(train)
print(pic)
cov = np.cov(train_id1[:,2], rowvar=0)
cov = np.cov(train_id1, rowvar=0)
\\
gau = [[None]*3]*3
gds = []
for i,X in enumerate([train_id1,train_id2,train_id3]):
    mean = np.mean(X, axis=0)
    cov = np.cov(X, rowvar=0)

    gau[i][0]=multivariate_normal(mean=mean[1],cov = cov[1,1])
    gau[i][1]=multivariate_normal(mean=mean[2],cov = cov[2,2])
    gau[i][2]=multivariate_normal(mean=mean[3],cov = cov[3,3])

    print ("[{:.4f} \quad {:.4f} \quad {:.4f}]".format(mean[1],mean[2],mean[3]))
    print ("[{:.4f} \quad {:.4f} \quad {:.4f}]\\\\"\
    .format(cov[1,1],cov[2,2],cov[3,3]))

wrong = 0
for i,v in enumerate(test):
    class1 = pic[0]*gau[0][0].pdf(v[1])*gau[0][1].pdf(v[2])*gau[0][2].pdf(v[3])
    class2 = pic[1]*gau[1][0].pdf(v[1])*gau[1][1].pdf(v[2])*gau[1][2].pdf(v[3])
    class3 = pic[2]*gau[2][0].pdf(v[1])*gau[2][1].pdf(v[2])*gau[2][2].pdf(v[3])
    c = np.array([class1,class2,class3])

    c_pred = np.argmax(c)
    c_true = -1
    if v[0]>=18 and v[0]<40:
        c_true = 0
```

```
    if v[0]>=40 and v[0]<60:
        c_true = 1
    if v[0]>=60:
        c_true = 2
    if c_true != c_pred:
        wrong = wrong+1
wrong = wrong /len(test)
```

Problem 4

```
#Questao 4
id1_t,id2_t,id3_t = [],[],[]
train_z = train.copy()
test_z = test.copy()

for i,v in enumerate(test):
    if v[0]>=18 and v[0]<40:
        id1_t.append(i)
    if v[0]>=40 and v[0]<60:
        id2_t.append(i)
    if v[0]>=60:
        id3_t.append(i)

train_z[id1,0]=1
train_z[id2,0]=2
train_z[id3,0]=3
test_z[id1_t,0]=1
test_z[id2_t,0]=2
test_z[id3_t,0]=3

gau = [None]*3
gds = []
for i,X in enumerate([train_id1,train_id2,train_id3]):
    mean = np.mean(X[:,1:], axis=0)
    cov = np.cov(X[:,1:], rowvar=0)

    gau[i]=multivariate_normal(mean=mean,cov = cov)

    print("Class",i+1)
    print ("[{:.4f} \quad {:.4f} \quad {:.4f}]".format(mean[0],mean[1],mean[2]))
    for i in cov:
        print ("{:.4f} & {:.4f} & {:.4f} \\\\".format(i[0],i[1],i[2]))

vo2 = np.zeros(len(test_z))
x_min, x_max = train[:, 3].min() - 2, train[:, 3].max() + 2
x = np.arange(x_min,x_max,0.25)

for i,v in enumerate(test_z):

    k = v[0].astype(int)
    vo2[i] = pic[k-1]*gau[k-1].pdf(v[1:])
    norm = 0
    for dx in x:
        norm = norm + pic[k-1]*gau[k-1].pdf([v[1],v[2],dx])
```

```
    vo2[i]=vo2[i]/norm


plt.scatter(test_z[:,3],vo2)
plt.grid(True)
plt.xlabel("Values of VO2max on test set")
plt.ylabel("Corresponding probability of this VO2max on test set")
plt.title("Probabilities of VO2max coditioned on the other features (test set)")

vo2 = np.zeros(len(test_z[:10]))
x_min, x_max = train[:, 3].min() - 2, train[:, 3].max() + 2
x = np.arange(x_min,x_max,0.25)
wrong = 0
for i,xi in enumerate(test_z[:10]):

    x3 = np.zeros([x.shape[0],3])
    for j,v in enumerate(x):
        x3[j]=[xi[1],xi[2],v]

    k = xi[0].astype(int)
    y = pic[k-1]*gau[k-1].pdf(x3)

    #plt.plot(x,y)
    vo2[i] = x[np.argmax(y)]

y_true = vo2.copy()
y_pred = test_z[:10,3].copy()
MSE = (((y_true - y_pred)**2).sum())/len(y_true)
print ("Generated:", vo2,"\n","True labels:",  test_z[:10,3], "MSE",MSE)
```

Problem 5

```
#Questao 5
from sklearn import mixture
import matplotlib as mpl
import itertools
X=train[:,1:]
lowest_bic = np.infty
bic = []
n_components_range = range(1, 7)
cv_types = ['spherical', 'tied', 'diag', 'full']
for cv_type in cv_types:
    for n_components in n_components_range:
        # Fit a Gaussian mixture with EM
        gmm = mixture.GaussianMixture(n_components=n_components,
                                      covariance_type=cv_type)
        gmm.fit(X)
        bic.append(gmm.bic(X))
        if bic[-1] < lowest_bic:
            lowest_bic = bic[-1]
            best_gmm = gmm

bic = np.array(bic)
```

```
color_iter = itertools.cycle(['navy', 'turquoise', 'cornflowerblue',
                              'darkorange'])
clf = best_gmm
bars = []

plt.figure(figsize=(8, 6))
spl = plt.subplot(2, 1, 1)
for i, (cv_type, color) in enumerate(zip(cv_types, color_iter)):
    xpos = np.array(n_components_range) + .2 * (i - 2)
    bars.append(plt.bar(xpos, bic[i * len(n_components_range):
                                  (i + 1) * len(n_components_range)],
                        width=.2, color=color))
plt.xticks(n_components_range)
plt.ylim([bic.min() * 1.01 - .01 * bic.max(), bic.max()])
plt.title('BIC score per model on the training set without \'idade\' features')
xpos = np.mod(bic.argmin(), len(n_components_range)) + .65 +\
    .2 * np.floor(bic.argmin() / len(n_components_range))

plt.text(xpos, bic.min() * 0.97 + .03 * bic.max(), '*', fontsize=14)
plt.grid(True,alpha=0.3)
spl.set_xlabel('Number of components')
spl.legend([b[0] for b in bars], cv_types)

gmm = mixture.GaussianMixture(n_components=3,covariance_type='full')
gmm.fit(train[:,1:])
y_train_pred = gmm.predict(train[:,1:])

plt.figure(figsize=(13,6))
for i in np.arange(4):
    plt.subplot(2,2,i+1)
    plt.scatter(train[:,i],y_train_pred,s=13, alpha=0.4)
    plt.xlabel(names[i],fontsize=13)
    plt.ylabel("Clusters",fontsize=13)
    plt.yticks([0,1,2])

plt.suptitle("Problem 5 mixture of gaussians with 3\
 clusters, checking for features",y=1.02,fontsize=14)
plt.tight_layout()
```

Problem 6

```
#Questao 6
gmm = mixture.GaussianMixture(n_components=3,covariance_type='full')
gmm.fit(train)

for (weight,mean,cov) in zip(gmm.weights_,gmm.means_,gmm.covariances_):
    print (weight)
    print ("[{:.4f} \quad {:.4f} \quad {:.4f} \quad \
    {:.4f}]".format(mean[0],mean[1],mean[2],mean[3]))
    for i in cov:
        print ("{:.4f} & {:.4f} & {:.4f} & {:.4f} \\\\"\
        .format(i[0],i[1],i[2],i[3]))
    print ('\n')
```

```
xi = np.array([81.5,181,32,6])
vo2 = np.zeros(len(test_z[:10]))

x_min, x_max = train[:, 0].min() - 2, train[:, 0].max() + 2
x = np.arange(x_min,x_max,0.005)

n = np.zeros([x.shape[0],4])
d = np.zeros([x.shape[0],4])

for j,v in enumerate(x):
    if 40<=v and v<50:
        d[j] = [v,xi[0],xi[1],xi[2]]
    n[j]=[v,xi[0],xi[1],xi[2]]

dem = np.sum(np.exp(gmm.score_samples(d)))
norm = np.sum(np.exp(gmm.score_samples(n)))
#print (n.shape,d.shape,norm.shape,dem.shape, dem/norm)
print("40<=v and v<50", dem/norm)

n = np.zeros([x.shape[0],4])
d = np.zeros([x.shape[0],4])

for j,v in enumerate(x):
    if 50<v and v<60:
        d[j] = [v,xi[0],xi[1],xi[2]]
    n[j]=[v,xi[0],xi[1],xi[2]]

dem = np.sum(np.exp(gmm.score_samples(d)))
norm = np.sum(np.exp(gmm.score_samples(n)))
print("50<v and v<60", dem/norm)

n = np.zeros([x.shape[0],4])
d = np.zeros([x.shape[0],4])

for j,v in enumerate(x):
    if 60<=v and v<=70:
        d[j] = [v,xi[0],xi[1],xi[2]]
    n[j]=[v,xi[0],xi[1],xi[2]]

dem = np.sum(np.exp(gmm.score_samples(d)))
norm = np.sum(np.exp(gmm.score_samples(n)))
print("60<=v and v<=70", dem/norm)
```

Problem 7

```
#Questao 7
from sklearn.cluster import KMeans
random_state = 170
X = data[:,[0,2,3]]
kmean = KMeans(n_clusters=3, random_state=random_state)
kmean.fit(X)
labels = kmean.labels_
n_groups = 6
clust=np.zeros([6,3])
```

```
for j,xi in enumerate(X):

    v=xi[0]
    if 18<=v and v<30:
        clust[0][labels[j]]+=1
    if 30<=v and v<50:
        clust[1][labels[j]]+=1
    if 50<=v and v<60:
        clust[2][labels[j]]+=1
    if 60<=v and v<70:
        clust[3][labels[j]]+=1
    if 70<=v and v<80:
        clust[4][labels[j]]+=1
    if 80<=v and v<100:
        clust[5][labels[j]]+=1

plt.figure(figsize=(10,5))
index = np.arange(n_groups)
bar_width = 0.25
opacity = 1
print (len(index),len(clust[:,0]))

rects1 = plt.bar(index-bar_width, clust[:,0],\
 bar_width,alpha=0.4,color='b',label='Cluter 0',hatch="//")

rects2 = plt.bar(index , clust[:,1],\
 bar_width,alpha=1,color='g',label='Cluter 1',hatch="\\")

rects3 = plt.bar(index + bar_width, clust[:,2],\
 bar_width,alpha=0.2,color='r',label='Cluter 2',hatch="o")

plt.xticks(index,labels_sectors)
plt.ylabel("number of clusters")
plt.xlabel("'idade 'brackets")
plt.title("K-means using 'idade', 'carga' and 'VO2max'")
plt.legend(loc='upper right',prop={'size': 20})
#For k-means with 4 clusters just re-do the above code with n_clusters=4
```

References

[1] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.