Grenoble INP - Ense3

AI and Autonomous systems

Lab 5 - M2-MARS

Linear models : Lasso Regularization

Author :
Souhaiel BEN SALEM

15 November 2021

Lab Objectives: The objective of this lab is to experiment with Lasso ($l1$) regularization and its use cases especially for sparsity promotion and to compare it with ridge regularization ($l2$) that we saw during the last lab.

Introduction:

While L2 regularization is an effective means of achieving numerical stability and increasing predictive performance, it does not address another problem with Least Squares estimates, parsimony of the model and interpretability of the coefficient values. While the size of the coefficient values is bounded, minimizing the RSS with a penalty on the L2-norm does not encourage sparsity, and the resulting models typically have non-zero values associated with all coefficients. L1 regularization has many of the beneficial properties of L2 regularization, but yields sparse models that are more easily interpreted

1 Notebook 1: Regularization for Sparsity: L1 Regularization (lasso)

1. **How does switching from $L2$ to $L1$ regularization influence the delta between test loss and training loss?**
   We notice that the the difference of the test loss and training loss becomes higher after going from L2 to L1. Indeed, $\Delta Loss = test\ loss - training\ loss$ went from 0.065 to 0.017
   after 500 epochs when using a regularization coefficient equal to 0.1 and from $0.046 to 0.013$ when using a regularization rate equal to 0.3. This means that our linear model performed better when used with L1 regularization (this can also be deduced by looking at the test error when using each of the norms $l1$ and $l2$).

2. **How does switching from $L2$ to $L1$ regularization influence the learned weights?**
   When using the L2 norms, we can see that weights are shrinked heavily but no component is eliminated (**no 0 weights**) as the model puts the most weight on the X1X2 feature. However, when using L1 regularization we notice that all the features are eliminated (**0 weights**) except for the X1X2 feature. In this case our model performed a the classification relying only on the **one** feature it deemed necessary. This is the main idea behind the L1 norm which promote sparsity by eliminating "unnecessary" features comlpetely and relying only on the informative ones.

3. **How does increasing the $l1$ regularization rate (lambda) influence the learned weights?**
   Increasing the regularization rate λ of the L1 norm makes the weight on the only features we got (X1X2) even smaller from $0.21 to 0.16$.

4. **Why the $l1$ penalty seems most appropriate than $l2$ one for this problem?**
   L1 penalty seems more appropriate for this problem since it provides a better test error, simpler, more interpretable model (and thus converges faster). Also L1 norm optimizes the median and therefore it is not sensitive to outliers.

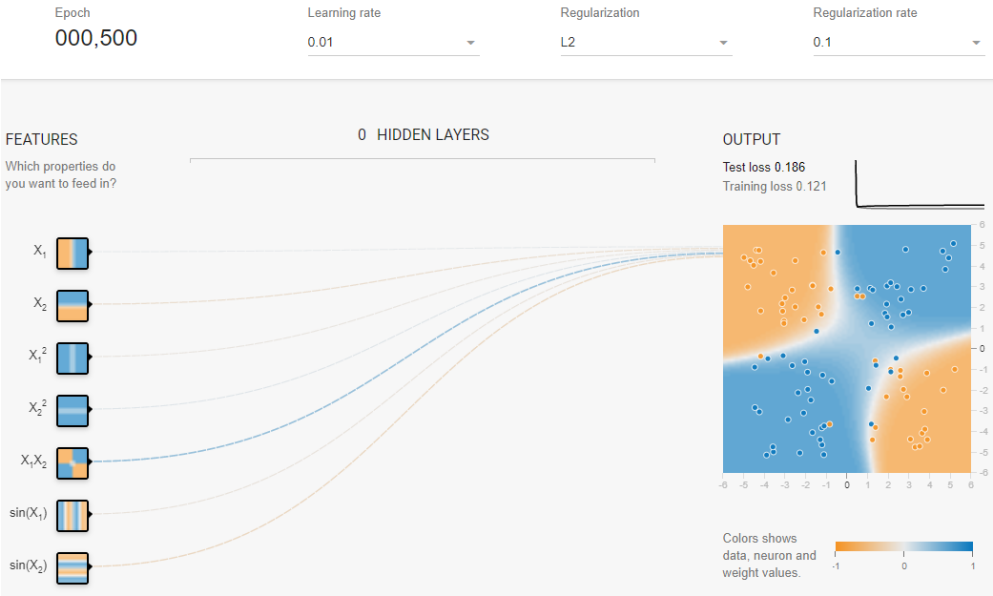

Figure 1: the results obtained after 500 when using L2 and $\alpha = 0.1$

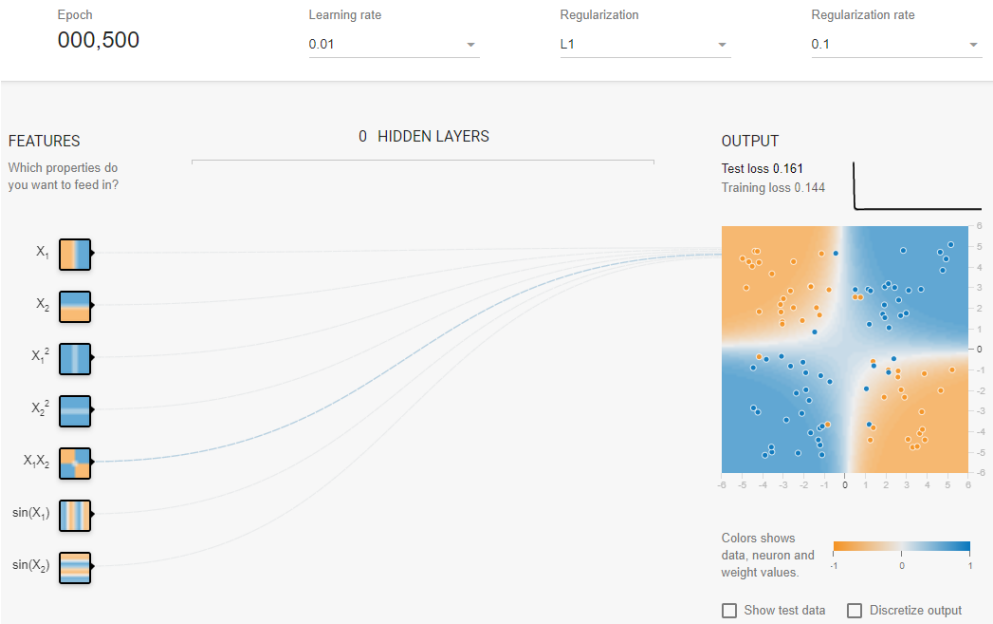

Figure 2: the results obtained after 500 when using L1 and $\alpha = 0.1$

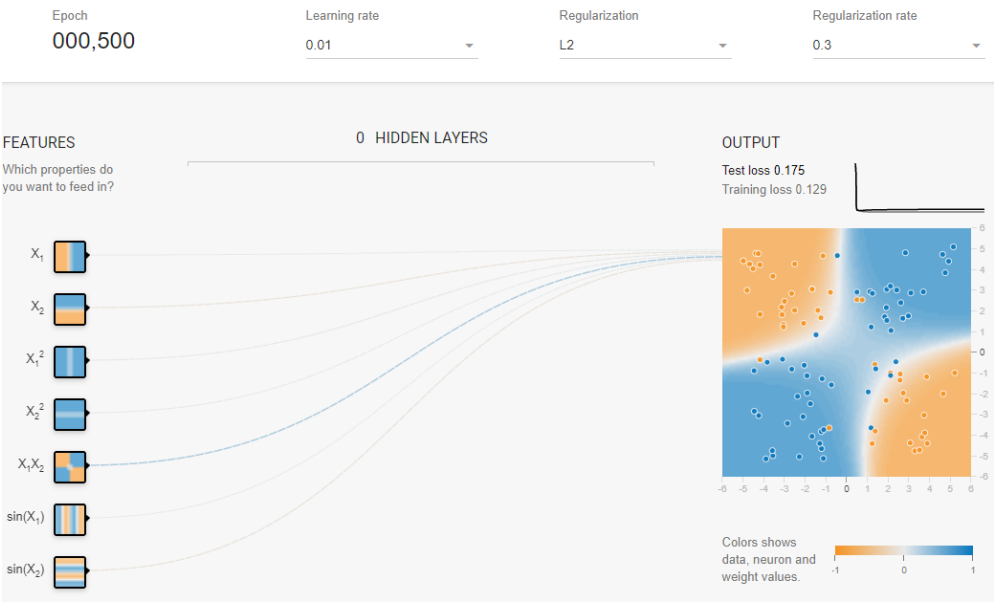

Figure 3: the results obtained after 500 when using L2 and $\alpha = 0.3$

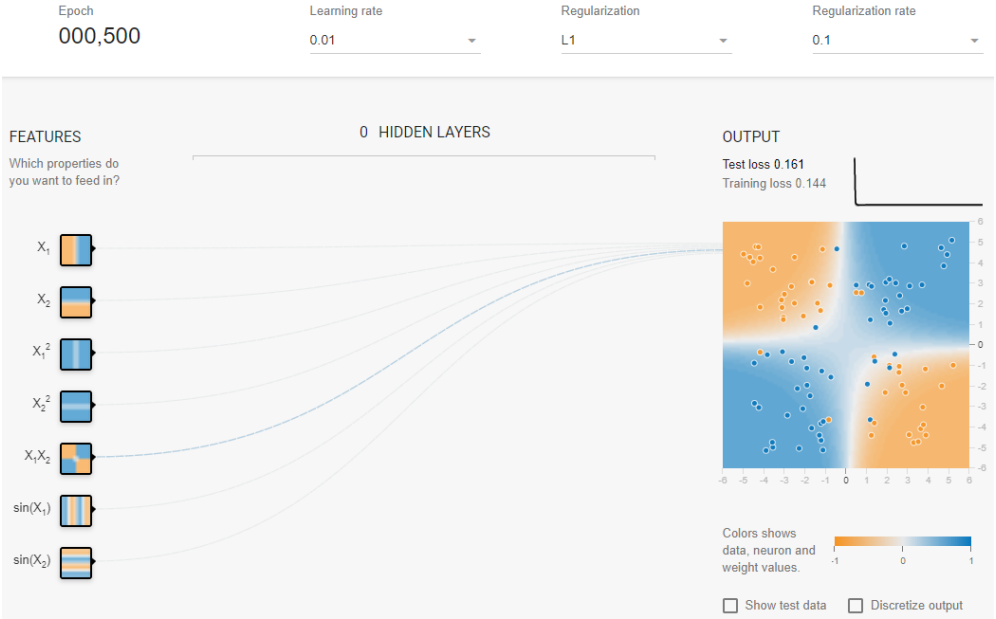

Figure 4: the results obtained after 500 when using L1 and $\alpha = 0.3$

1.1 Task 2:

1. **How does introducing $L1$ regularization influence the test loss and also the delta between test loss and training loss?**
   We notice that introducing the $l1$ regularisation improves the test loss (it went from 0.430 to 0.343 in our experiment), worsen the training loss (it went from 0.142 to 0.292 in our experiment) and as a result of the previous observations , the delta between the test loss and training loss decreased ($\Delta Loss = test\ loss - training\ loss$ went from 0.288 to 0.051). Introducing the $l1$ regularization increases the bias and decreases variance.

2. **How does introducing $L1$ regularization influence the learned weights?**
   We notice that after introducing the $L1$ regularization, most of the weights are equal to 0. In other words, most of the neurons were eliminated and we got a model that it much more simpler that the one we got without using regularization.
   After 1000 epoch our model is using exactly 3 neurons from the input layer,only one neuron from each hidden layer and one neuron from the output layer.

3. **What are the only features that are selected with $L1$ regularization? Is it in agreement with the 'optimal' decision boundary for this data set? Are the hidden layers useful here?**
   The only features selected by the $L1$ regularization are $X1^2$ and $X2^2$ which is in total agreement with the expected decision boundary which is a **circle**. In this case it appears that the hidden neurons are not necessary as the decision boundary can be derived from the just the input features.

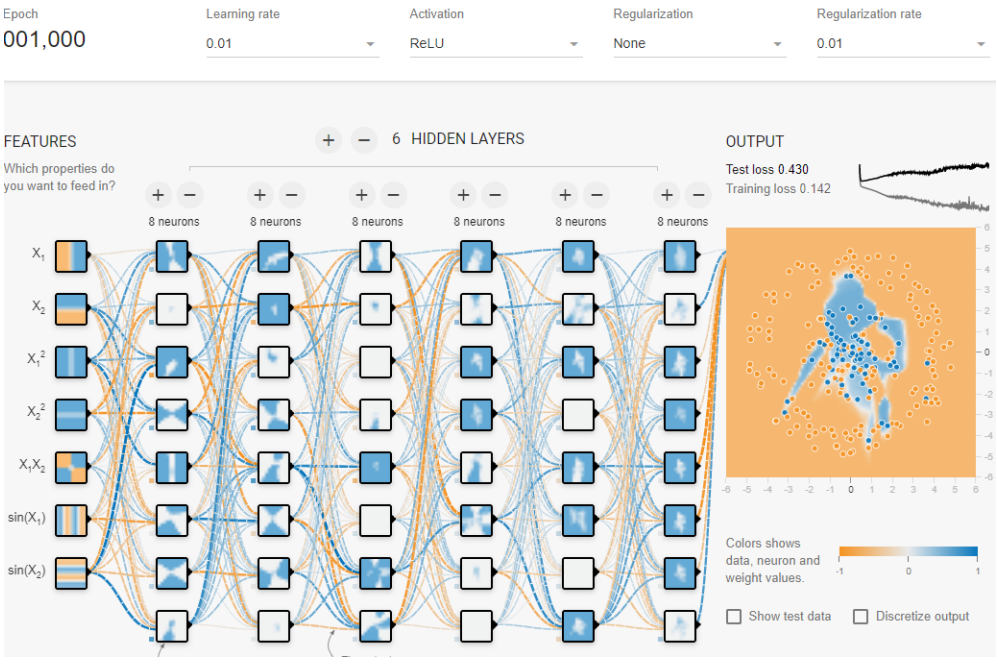

Figure 5: the results obtained after 1000 without using L1 regularization

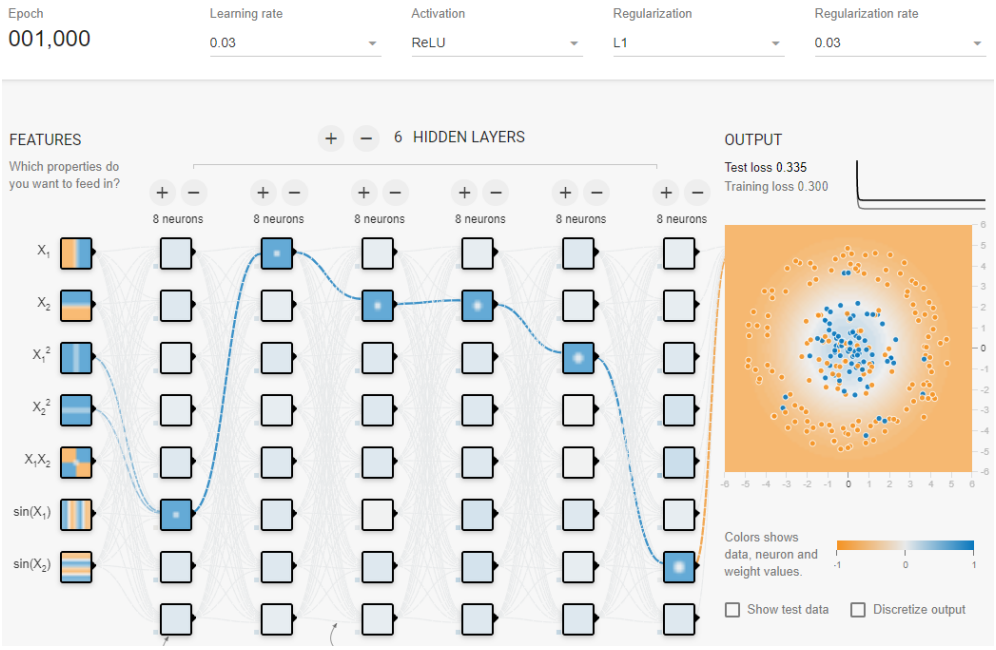

Figure 6: the results obtained after 10000 when using L1 and $\alpha = 0.03$

2 Notebook 2: South Africa Heart Diseases Data

2.1 Compute ordinary (without regularization) Logistic Regression:

2.1.1 Exercise:

1. **what does a positive, negative or near-zero weight mean to predict heart disease?**
    - **positive weight**: A positive weight represents an excitatory connection which means that the existence of the feature associated with this weight contributes to the occurrence of the output (the patient being diagnosed wit CHD in this case).
    - **zero or near-zero weight**: means that the considered variable (feature) barely or does not contribute at all in the prediction of the output (the observed variable is not informative of the output variable).
    - **negative weight**: A negative weight represents an inhibitory connection between the input and output. In other words negative weights reduce the value of the output which means that the variable associated with such weight decreases the chance of observing the output (in this case, it means that a variable associated with a negative weight reduces the chances of being diagnosed with CHD).

2. **How do you interpret the weight of obesity for instance?**
   Obesity is associated with a negative weight, which means that obese people have less chance of being diagnosed with CHD (or the more obese you are, the less likely you will be diagnosed with CHD), which makes no sense.
3. **How can you explain such surprising findings?**
   This absurd, unexpected result is probably associated with the fact that we applied OLS directly without applying regularization.

2.2 Compute ℓ1 penalized Logistic Regression and lasso path

2.2.1 exercise:

1. **What are the only significant variables estimated with cross-validation?**
   The only significant variables (variables with non-zero coefficients) selected by the $L1$ regularization after cross-validation are tobacco, ldl, famhist, typea and age, with age being the most significant.
2. **How can we rank them by significance order (hint: look at the lasso path)?**
   The most significant variables are, from most to least significant: age, famhis, tobacco, ldl and finally typea.
3. **Do these results seem more credible (than those obtain without regularization) to predict heart diseases?**
   Yes these results seem more credible since the variable selected by regularization are logically sorted by significance and there are no unexpected coefficients.

2.3 Compute the predicted CHD probability for some patients

2.3.1 Exercise:

1. **Based on the logistic regression formula to compute the probability of each class (with, or without CHD) and the values of the estimated weights, what would be the increase factor on the odds probability when the tobacco consumption increases of 1 (in standardized unit)?**
   The logistic regression model output is defined as

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + ..\beta_n X_n$$

   For our sparse model, this reduces to:

$$Y_i = intercept + \beta_{age} X_{age} + \beta_{famhist} X_{famhist} + \beta_{tobacco} X_{tobacco} + \beta_{ldl} X_{ldl} + \beta_{typea} X_{typea}$$

   To get the response for the first patient we just need to plug in the values of his standarized features into the formula of our sparse model.

Afterwards we get the probability of the patient beignosed bu CHD by using the probability formula:

$$P = \frac{1}{1 + e^{-y}}$$

And the Odds:

$$Odds = \frac{P}{1 - P}$$

The manual computation of the probability and odds is as follows:

```
import math
y_patient1=(betas_cv['tobacco'][0]+1)*x.iloc[0][1]+betas_cv['age'][0]*x.iloc[0][8]+betas_cv['ldl'][0]*x.iloc[0][2]+betas_cv['famh
prob_CHD=1/(1+math.exp(-y_patient1))
Odds=prob_CHD/(1-prob_CHD)
print(prob_CHD)
print(Odds)
```

Figure 7: Computation of the probability and odds using the formula

2. **Check this by adding an offset to the tobbaco variable in the cell above and comparing the obtained odds**
   The computation of the probability and odds using scikit-learn after the tobacco consumption increases by 1 is as fllows:

```
# Get the predicted risk for the first patient
ipatient = 0 # patient index
x = Xs[ipatient,:].reshape(1, -1)
x = pd.DataFrame.from_records(x, columns=names)
ylabel = 'Case' if y[ipatient] else 'Control'
print("{} Patient with (standardized) features:\n{}\n".format(ylabel, x))

# TODO: increase/decrease tobacco consumption
x_copy = x.copy()
x_copy['tobacco'] += 1 # offset to add in standardized unit

# Proba of heart disease
proba_CHD_0 = model.predict_proba(x)[0,1]
proba_CHD = model.predict_proba(x_copy)[0,1]
print("Proba of coronary heart disease (CHD): {:.3f}".format(proba_CHD))
print("Odds probability of CHD: {:.3f}".format(proba_CHD/(1-proba_CHD)))
```

```
Case Patient with (standardized) features:
        sbp   tobacco       ldl  adiposity   famhist    typea   obesity  \
0  1.058564  1.823073  0.478412  -0.295503  1.185854 -0.41847 -0.176786

    alcohol       age
0  3.277738  0.629336

Proba of coronary heart disease (CHD): 0.650
Odds probability of CHD: 1.858
```

Figure 8: The probability and odds after the tobacco consumption increases by 1

2.4 Greedy variable selection procedure

2.4.1 Exercise:

1. **for our heart diseases dataset, usig a $K5$ =-fold CV, how many fits would be required to find the best subset?**
   For our problem, we have $p = 9$ variable, so if we use K-fold cross validation, we need to fit $K \times 2^p = 5 \times 2^9$ models.

2.5 Backward selection: Recursive Feature Elimination (RFE)

2.5.1 Exercise:

1. **Change the desired number of features to 1 to rank the features by significance order (most significant variables must enter first in the model). Is it perfectly consistent with the Lasso path ranking?**
   After changing the number of desired features to 1, the selected variable is **age**. In fact, we got the same ranking of significance as the one obtained with Lasso regression:

```
This took 0.035s
The selected variables are: ['age']
```

	age	famhist	tobacco	typea	ldl	obesity	adiposity	sbp	alcohol
Rank	1	2	3	4	5	6	7	8	9

Figure 9: The ranking of variables given by RFE

2. **What is hyperparameter to optimize for the RFE procedure? How can you estimate it?**
   The hyperparameter to be optimized in the case of RFE is the number of significant features to keep. We can optimize this parameter using cross-validation.
3. **Replace the RFE procedure by feature$_s$election.RFECV toselectthemostsignificantvariables.Isitinagreementwiththe Lassoresults?**
   After replacing the RFE procedure by the RFECV 5-fold cross validation, the selected, most significant variables are age, famhis, tobacco, ldl and finally typea, which is exactly the result we got with the Lasso regularization.
4. **Do you think that this procedure is still appropriate when the number of variables p is greater than the sample size n of the training set?**
   Yes, this technique would be appropriate for high dimensional problems because the algorithm would select the subset of the most significant features reducing the dimensions of the problem to only that of the selected subset and allowing us to avoid overfitting.

```
from sklearn.feature_selection import RFE, RFECV
clf = LogisticRegression(penalty='none', tol=1e-6, max_iter=int(1e6))
start = time()
selector = RFECV(clf, step=1,cv=5)
selector = selector.fit(Xs, y)
print("This took %0.3fs" % (time() - start))
print("The selected variables are: {}".format(np.array(names)[selector.support_]))
sel_ind_sorted = np.argsort(selector.ranking_)
ranking = pd.DataFrame.from_records(selector.ranking_[sel_ind_sorted].reshape(1,p),
                                    columns=[names[i] for i in sel_ind_sorted], index=['Rank'])
ranking.head()
```

```
This took 0.136s
The selected variables are: ['tobacco' 'ldl' 'famhist' 'typea' 'age']
```

	tobacco	ldl	famhist	typea	age	obesity	adiposity	sbp	alcohol
Rank	1	1	1	1	1	2	3	4	5

Figure 10: The result obtained after applying the RFECV cross validation

5. **Change the desired number of variables (number of nonzero coefs) from p to 1 to rank the features by signicance order (most significant variables must enter first in the model). Is it perfectly consistent with the previous results?** After changing the number of desired features to 1, the selected variable is **age**. In fact, we got the same ranking of significance as the one obtained with the previous results from RFE and Lasso regression.

6. **What is hyperparameter to optimize for the OMP procedure? How can you estimate it?**
The hyperparameter to be optimized in the case of OMP is the number of non-zero coefficients. We can optimize this parameter using cross-validation.

7. **Replace the OMP procedure by**
linear$_{m}$*odel.OrthogonalMatchingPursuitCV toselectthemostsignificantvariables.Isitinagreementwiththepreviousres*
After replacing the OMP procedure by the rthogonalMatchingPursuitCV 5-fold cross validation, the selected, most significant variables are age, famhis, tobacco, ldl and finally typea, which is exactly the result we got with RFE and the Lasso regularization.

```
# We use Forward selection, aka Orthogonal Matching Pursuit
from sklearn.linear_model import OrthogonalMatchingPursuit, OrthogonalMatchingPursuitCV

# Get the 5 most significant features
omp = OrthogonalMatchingPursuitCV(cv=5,fit_intercept=True, normalize=True)
omp.fit(X, y)
coef = omp.coef_
idx_r, = coef.nonzero()
sel_feat_name= [names[l] for l in idx_r] ;
print("Most significant features: {}".format(np.array(names)[omp.coef_.nonzero()]))
```

```
Most significant features: ['tobacco' 'ldl' 'famhist' 'typea' 'age']
```

Figure 11: The result obtained after applying the OrthogonalMatchingPursuitCV cross validation

8. **Compared to greedy methods that are suboptimal procedures to approximate the best sparse solution (the best subset), Lasso penalty is often presented as a convex relexation of the sparsity constraint. Can you explain why? What are the possible benefits/disadvantages of the greedy and lasso procedures?**
   Lasso regularization yields the optimal sparse solution because it corresponds to minimize a convexe loss function $(Y - X\beta)^T(Y - X\beta) + \lambda|\beta|_1$. (This is true when the rank of the data matrix is of full rank i.e $rank(X) = p$ because the criterion is strictly convex).
   However, greedy methods converge to a sub-optimal solution i.e a local minimum, so they may yield a sparse solution but that solution might not be the optimal (best) one.

3 Notebook 3: Lasso and the curse of dimensionality

3.1 linear regression on the original dataset

3.1.1 Exercise:

1. **Does the linear model seems accurate for this task and dataset?**
   After standardization, our model was able to explain 88% of the variance in the data. The "acceptable" accuracy depends on the problem but here with only 3 features and 200 samples we should be able to predict the sales more precisely. So, maybe another model would have been more accurate.

2. **What are the most significant variables to predict scale (hint: you may scale the training data if you want to be sure to correctly interpret the weight)**
   The most significant variables are, from most to least significant, **TV**, **Radio and Newspapers**.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Split data -> 1/3 for learning & 2/3 for validation
X_std=scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_std, y, test_size=0.75, random_state=0)
print('Training set dimensions: {}\n'.format(X_train.shape)) # training set dimensions: n (sample size) x p (dimension size)
sc = StandardScaler()
#Xs= sc.fit_transform(X_train)
#Ys=sc.fit_transform(X_test)
# Apply linear regression
from sklearn.linear_model import LinearRegression
linrgr =  LinearRegression()
linrgr.fit(X_train, y_train)

# Get the estimated weights
df_weights = pd.DataFrame.from_records( linrgr.coef_.reshape(1,-1), columns=X.columns, index=['Estimated Weights'])
print(df_weights)

# Get the coefficient of determination R2
r2 = linrgr.score(X_test,y_test)
print("\nLinear model explains {:.0f}% of the Sales variance".format(100*r2))
```

```
Training set dimensions: (50, 3)

                       TV     Radio  Newspaper
Estimated Weights  3.711107  3.067802   0.173484

Linear model explains 88% of the Sales variance
```

Figure 12: The result obtained after Standardization

3.2 Curse of dimentionality

3.2.1 Exercise:

1. **Does the methods without regularization work well on this high dimensional and noisy data?**
   most of these methods (SVC, K-NN and Linear regression) seem to perform poorly when applied without regularization on this noisy dataset except for the random forest which yields a kind of acceptable result especially when compared to other methods. This is because random forests randomly selects observations/rows and specific features/variables to build multiple decision trees from and then averages the results which mitigate the curse of dimentionality for a certain extent.
2. **Does the knn regressor based on euclidean metric seems accurate for this task and dataset? How can we explain that?**
   The K-NN regressor does not seem appropriate of this task since the coefficient of determination does not exceed 0.3 for the test data which means that the algorithm was only able to explain 30% of the variance in the data. K-NN is known to be a bad choise for high dimentional, noisy datasets.The algorithm will try to find k nearest neighbours but all points are random. Also K-NN is very sensitive to outliers which makes it even prone to misclassifications.

3.3 Regularization and Variable selection

3.3.1 Exercise:

1. **Does the lasso regression works well on this high dimensional and noisy data?**
   The lasso regression seems to perform well on the noisy dataset especially when compared with the previous methods as it was able to explain 88% of the variance in the data.
2. **Did it succeed in recovering the significant features?**
   Yes, the lasso regression was able to recover only the most significant features which are (as confirmed at the start of the notebook) **TV and Radio**. It was however slightly affected by noise_19.
3. **What are the two most significant variables according to the Lasso Path?**
   According to the Lasso path, the most significant variables are **TV** and **Radio**. This can be confirmed if we perform a k=5-fold cross validation to determine beta and lambda:

```
from sklearn.linear_model import LassoCV
from time import time
print("Computing K-fold CV ...")
# K fold cross validation  (K=5)
start = time()
model =LassoCV( cv=5,alphas=alphas, random_state=0).fit(X_train,y_train)
print("This took %0.3fs" % (time() - start))
```

```
Computing K-fold CV ...
This took 0.052s
```

```
# Now model is tuned with the penalty parameter estimated by CV
lambda_cv = model.alpha_
# The coef estimated with CV
beta_l1norm = np.sum(np.abs(model.coef_))

print('CV estimates:')
print('- lambda = {:.3f}, which yields ||beta||_1 = {:.3f}\n'.format(lambda_cv,beta_l1norm) )
```

```
CV estimates:
- lambda = 0.388, which yields ||beta||_1 = 6.196
```

Figure 13: The optimal lambda and beta obtained by cross-validation

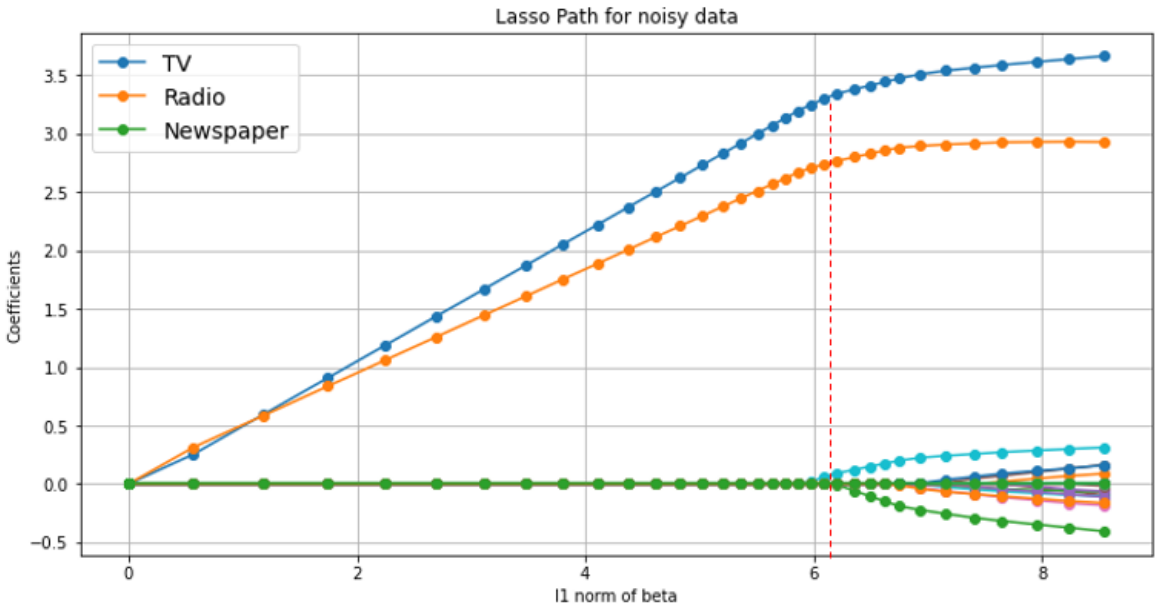

Figure 14: Confirming the result using the Lasso path

Conclusion:

During this lab we experimented with the $l1$ and its use case for linear model as we tackled both regression and classification problems. We verified that the Lasso is a great tool for sparse linear regression, especially for problems in which the number of variables p exceeds the number of observations n. But when p > n, the lasso criterion is not strictly convex, and hence it may not have a unique minimizer. An important question is: how can we manage the case of non-uniqueness in lasso solutions? This question is to be answered in future explorations.