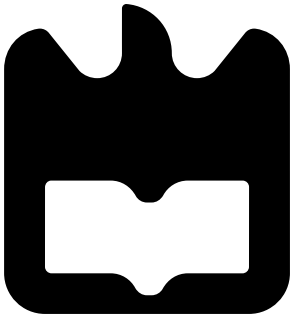
UNIVERSITY OF AVEIRO

Department of Electronics, Telecommunications and Informatics

Foundation of Machine Learning

Prof. Petia Georgieva



Regression problem
Prediction of final grades

Group members:

Stefano Petrina - (112056)

Stanisław Franczyk - (112059)

Aveiro November 2022

Contents

1 Data description and preprocessing

For the first project of the subject Foundation of Machine Learning selected set **Alcohol Effect on Courses Grades**[1] from *kaggle* dataset.

1.1 Data description

This data approach student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school related features and it was collected by using school reports and questionnaires. Data descriptions are shown in Table 1.

This model represent an example of multivariate linear regression. The parameter to predict is the final grade. The choice of this model was dictated by the fact that the issue of how the habits of a student affect his final grade is very easy to understand but very difficult to execute.
In this model there is a large amount of different data about different habits: thanks to that it is possible to try to remove some data to see how the result obtained change using different combinations of parameters.

1.2 Overfitting problem

The choice of a combination of data is also due to the risk of overfitting: using all the given parameters, the model may become too complex. When the model memorizes irrelevant information, it fits too closely to the training set and it becomes “overfitted”, and it is unable to generalize well to new data. If a model cannot generalize well to new data, then it will not be able to perform the classification or prediction tasks that it was intended for.

1.3 Preprocessing

Before solving the regression problem all data from the set should be prepared. For the correct working algorithm, all given data must be numerical type. If they aren’t, all non-numeric features are selected. Next, if the feature is binary

[1]The source: https://www.kaggle.com/datasets/whenamancodes/alcohol-effects-on-study

Table 1: Descriptions of features

Columns	Description
school	student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
sex	student's sex (binary: 'F' - female or 'M' - male)
age	student's age (numeric: from 15 to 22)
address	student's home address type (binary: 'U' - urban or 'R' - rural)
famsize	family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
Pstatus	parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
Medu	mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)
Fedu	father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)
Mjob	mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at home' or 'other')
Fjob	father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at home' or 'other')
reason	reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
guardian	student's guardian (nominal: 'mother', 'father' or 'other')
traveltime	home to school travel time (numeric: 1 - ¡15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - ¿1 hour)
studytime	weekly study time (numeric: 1 - ¡2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - ¿10 hours)
failures	number of past class failures (numeric: n if $1 <= n < 3$, else 4)
schoolsup	extra educational support (binary: yes or no)
famsup	family educational support (binary: yes or no)
paid	extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
activities	extra-curricular activities (binary: yes or no)
nursery	attended nursery school (binary: yes or no)
higher	wants to take higher education (binary: yes or no)
internet	Internet access at home (binary: yes or no)
romantic	with a romantic relationship (binary: yes or no)
famrel	quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
freetime	free time after school (numeric: from 1 - very low to 5 - very high)
goout	going out with friends (numeric: from 1 - very low to 5 - very high)
Dalc	workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
Walc	weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
health	current health status (numeric: from 1 - very bad to 5 - very good)
absences	number of school absences (numeric: from 0 to 93)
G1	first period grade (numeric: from 0 to 20)
G2	second period grade (numeric: from 0 to 20)
G3	final grade (numeric: from 0 to 20, output target)

(for example it has only categories 'yes' and 'no') each category is changed to 1 or 0.

Else, if the feature has more than 2 categories it should be split into columns for each. If in the sample occurs the given category this column is set to 1, else set to 0.

The data preparation is called the preprocessing. The function that does preprocess was written for the project and is shown in Listing 1. The function take data type DataFrame. It use *LabelEncoder()* form *sklearn.preprocessing* library to convert binary columns and *get_dummies* from *pandas* library for other non-numerical cases. The *get_dummies* function works exactly as described above, creates new columns based on a set of categories inside a given column.

Listing 1: Converting categorical string data into numeric

```
1  def features_string_to_numeric(data):
2      """
3      Changes columns with string data to numeric data by creating a
           new column for each new string in the column.
4          Case of new column name: {column name}_{string data}
5      """
6      df = data.copy()
7      columns = df.select_dtypes('object')
8      le = LabelEncoder()
9
10     binary_columns = []
11     for c in columns:
12         if len(df[c].unique()) == 2:
13             df[c] = le.fit_transform(df[c])
14             binary_columns.append(c)
15     columns.drop(binary_columns, axis=1, inplace=True)
16
17     df1 = pd.get_dummies(columns, prefix=columns.columns)
18     df = pd.concat([df, df1], axis=1).reindex(df.index)
19
20     df.drop(columns.columns, axis=1, inplace=True)
21
22     return df
```

1.4 Feature Normalization

Feature normalization is one of the most useful pre-processing operations because it improves the quality of the final results. It allows to normalize the range of variation of the features of the dataset. That is very important because when the range of values is highly variable, the simple Euclidean distance between two points can become misleading.

This method reduces the time in which the learning algorithm converges to the final result and improves the effectiveness of the statistical model. The code used to accomplish this operation is in Listing 2.

Listing 2: Feature Normalization

```
1 def featureNormalization(X):
2     """
3     Take in numpy array of X values and return normalize X values,
4     the mean and standard deviation of each feature
5     """
6
7     mean = np.mean(X, axis=0)
8     std = np.std(X, axis=0)
9
10    X_norm = (X - mean) / std
11
12    return X_norm, mean, std
```

2 Data visualization

For the realization of the model, the relationship of interest is the one between G3 (final grade, i.e. the parameter to predict) and all the other parameters.

2.1 Relation with Math Grade

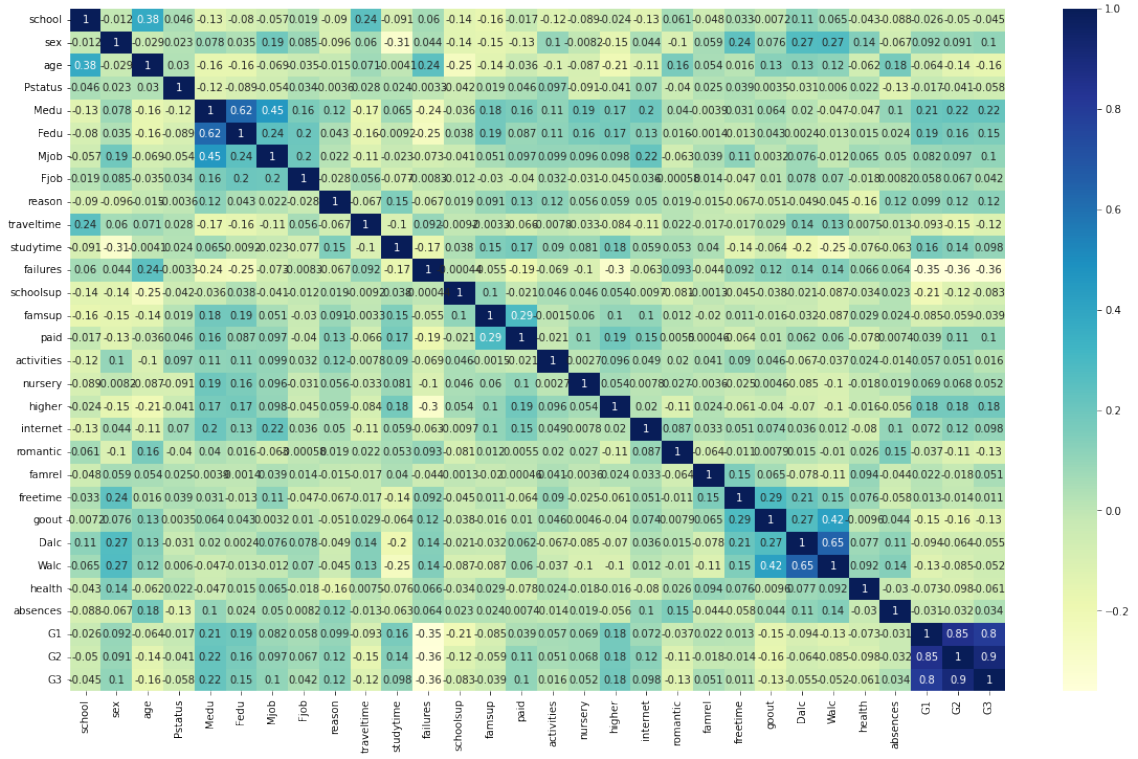


Figure 1: Heatmap

This is the heatmap that shows the relation between all the parameters regarding the Math grade. Looking at the figure 1, it is easy to observe that all the features have very less significant effect on the final grade, except for G1 and G2, that represent respectively the first and the second period grade. For this reason, G1 and G2 will not be used for the realization of the model

The goal of the table 2 is to focus the attention only on the relation between the grade of the math course and the other parameters.

Table 2: Correlation between G3 and other parameters for math data

Parameter	Value	Parameter	Value
G2	0.904868	famsize	0.081407
G1	0.801468	Mjob services	0.078429
failures	-0.360415	health	-0.061335
Medu	0.217147	Pstatus	-0.058009
higher	0.182465	Mjob teacher	0.057712
age	-0.161579	Fjob health	0.057111
Fedu	0.152457	Dalc	-0.054660
goout	-0.132791	Fjob other	-0.053483
romantic	-0.129970	reason other	0.052008
traveltime	-0.117142	Walc	-0.051939
Mjob health	0.116158	nursery	0.051568
Mjob at home	-0.115634	famrel	0.051363
address	0.105756	school	-0.045017
sex	0.103456	famsup	-0.039157
paid	0.101996	absences	0.034247
reason course	-0.098950	guardian father	0.032493
internet	0.098483	guardian mother	0.022338
studytime	0.097820	reason home	-0.021359
Mjob other	-0.096477	Fjob services	-0.016108
reason reputation	0.095692	activities	0.016100
Fjob teacher	0.095374	Fjob at home	-0.013385
guardian other	-0.087774	freetime	0.011307
schoolsup	-0.082788	-	-

2.2 Relation with Portuguese Grade

The values that represent the relationship between the final grade of the portuguese course and the other parameters are slightly different from the ones in table 2. Also in this case G1 and G2 have a value much higher than the others, and for the same reason as before they will not be used. Here there is the list of the 10 parameters with the higher value; they will be used later for the data splitting.

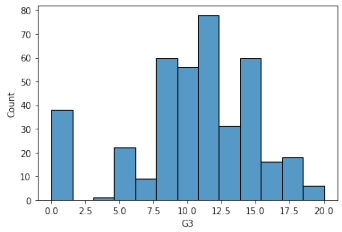
1. failures
2. higher
3. school
4. studytime
5. Medu
6. Fedu
7. Dalc
8. Walc
9. reason
10. address

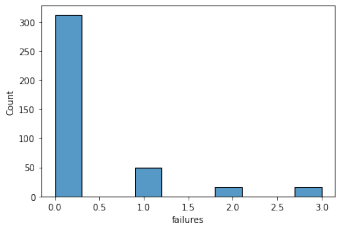
2.3 Histograms

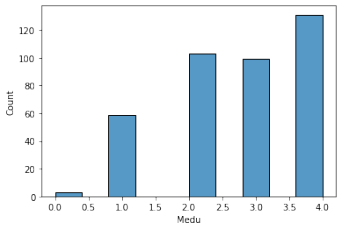
This section shows the histograms related to the most significant values in relation to the final grade and the histogram related to the final grade. In the first section the histograms related to the math course are shown, in the second those related to the Portuguese course.

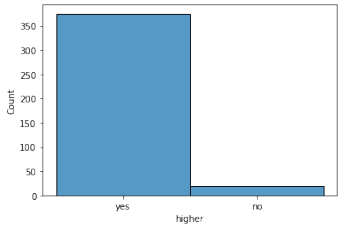
2.3.1 Histograms related to the math course

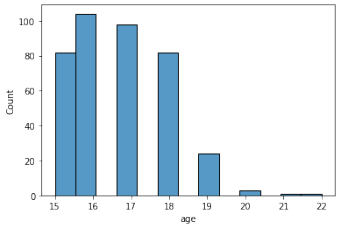
Final grade

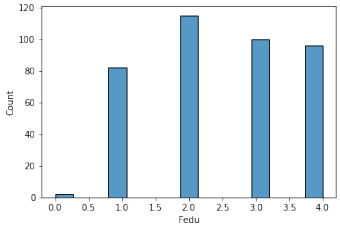


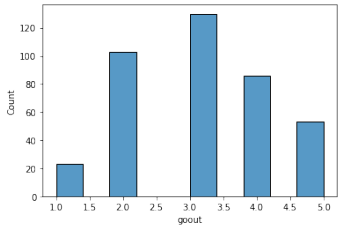


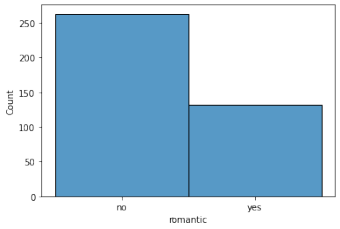


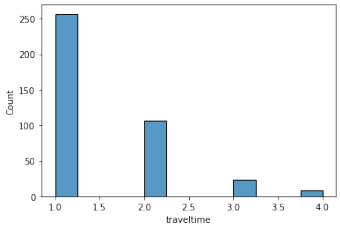


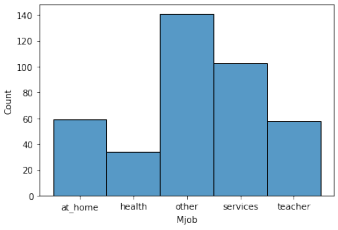


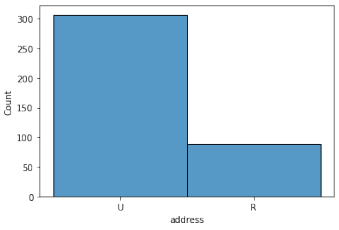






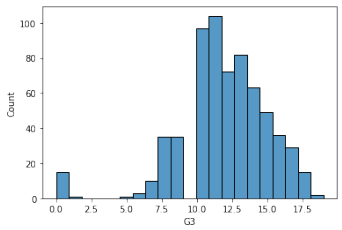


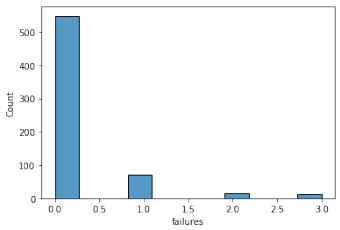


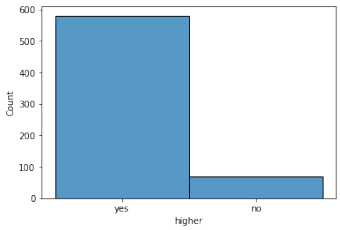


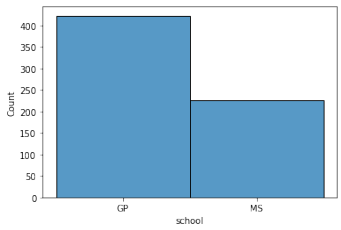
2.3.2 Histograms related to the Portuguese course

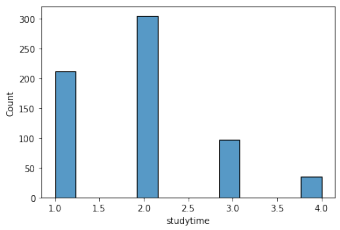
Final grade

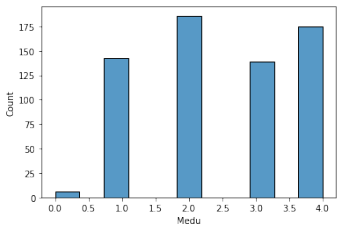


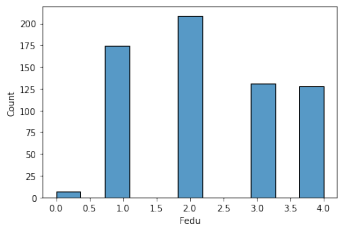


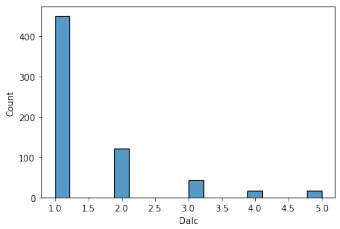


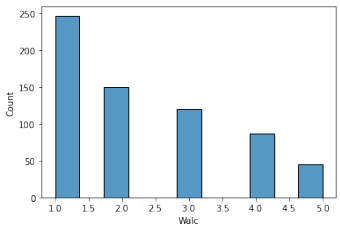

Count
GP
MS
school

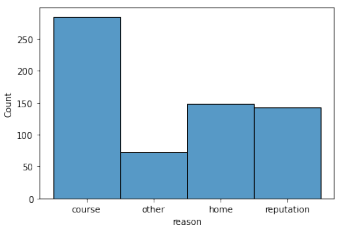

Count
studytime

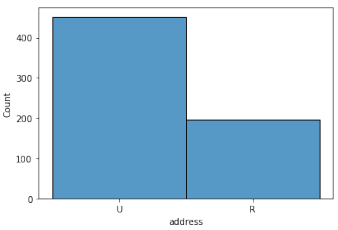

Count
Medu


Count
Fedu


Count
Dalc


Count
Walc


Count
course
other
home
reputation
reason


Count
U
R
address

3 Short description of the implemented ML models

Linear regression is creating a model from delivered data. Based on the data linear function is computed. The gradient descent method can be used for the best fit.

Regresslion linear model is written as follows:

$$h(\theta) = \theta^T \cdot x = \theta_0 + \theta_1 x_1$$

To assess the quality of the model, Mean Square Error is used. It is used inside each iteration of the gradient descent algorithm. MSE is written as follows:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_o(x^{(i)} - y^{(i)})^2$$

where m is number of training samples.

In order to achieve the goal of minimizing the cost function, the number of iterations is set, and then the quality improvement of the model is performed in a loop by improving θ. Before do it we we have to compute the cost function gradient:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_o(x^{(i)} - y^{(i)}) x_j^{(i)}$$

Then new θ can be computed.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

where α is hiperparameter that indicates how big a step the algorithm should make in one iteration. If it is too small for a limited amount of time, the algorithm will not find a global minima. Too large will cause the function to move away from the minimum. The most optimal allows you to find the minimum cost of the function.

4 Model training

4.1 K-Fold Cross Validation

Given the large number of samples observed, the cross validation technique is used to evaluate the implemented model, which consists of dividing the total data set (i.e. the students) into k parts of equal size. Once the partition has been made, one group at a time is iteratively excluded (test data) and an attempt is made to predict it with the non-excluded groups (training data), in order to verify the validity of the prediction model used. The function made to compute this technique is the following:

Listing 3: K-Fold Cross Validation

```
1 def Kfold(X, y, alpha, K=10, iteration_number=400):
2     """
3     Divide the data to do the K-fold
4     compute the gradient descent of each fold using gradientDescent
5     use computeCost for each fold
6     RETURN: average of all the compute costs
7     """
8
9     elements = int(X.shape[0] / K)
10    comp = 0
11
12    for i in range(0, K):
13        theta = np.zeros((n,1))
14
15        split_index1 = i * elements
16        split_index2 = split_index1 + elements
17
18        X_train, X_test = split_train_test(X, split_index1,
              split_index2)
19        y_train, y_test = split_train_test(y, split_index1,
              split_index2)
20
21        theta, J_history = gradientDescent(X_train, y_train, theta,
              alpha, iteration_number)
22
```

```
23          comp += computeCost(X_test, y_test, theta)
24
25      return comp / K
```

with

Listing 4: Function *split_train_test*

```
1 def split_train_test(data, a, b):
2     x_train1 = data[ :a, :]
3     x_test   = data[a:b, :]
4     x_train2 = data[b: , :]
5
6     x_train = np.vstack((x_train1, x_train2))
7
8     return x_train, x_test
```

4.2 Data Splitting

Given the large amount of parameters related to each student, it is possible to create the model by selecting only a part of the parameters. In this way the model will be less complex. In this case, the choice falls on the best 10 parameters, i.e. those with the highest value of the relationship with G3.

Referring to the math course, table 2 shows the relationship that each parameter has with G3, and this relationship can be used to simplify the model and perform the regression using only the most significant parameters. In this case, the 10 best values from the 2 table are selected, i.e. the following:

1. failures
2. Medu
3. higher
4. age
5. Fedu
6. goout

7. romantic
8. traveltime
9. Mjob
10. address

Referring to the Portuguese class instead, the 10 best parameters are:

1. failures
2. higher
3. school
4. studytime
5. Medu
6. Fedu
7. Dalc
8. Walc
9. reason
10. address

The code to select which parameters to consider is the following

```
1 def take_features_get_xy(data, col_take, col_y):
2     data2 = data[col_take]
3     data2 = features_string_to_numeric(data2)
4
5     y_df = data[col_y]
6     return data2.values, y_df.values, data2
```

While the code to decide which parameters to remove is the following

```
def drop_features_get_xy(data, col_drop, col_y):
    data2 = data.drop(col_drop, axis=1)
    data2 = features_string_to_numeric(data2)

    y_df = data2[col_y]
    X_df = data2.drop([col_y], axis=1)
    return X_df.values, y_df.values, X_df
```

The latter serves to remove the values of G1 and G2 to create the model that uses all the parameters.

4.3 Learning Rate

Once the k-fold cross validation function has been defined and after having decided how to organize the data splitting, the next phase consists in finding the best value to use as the learning rate, i.e. the α value which allows to obtain the lowest cost. If the learning rate is too small, the gradient descent takes a very long time to converge to the optimal value. If the learning rate is too large, $J(\Theta)$ can diverge and ”blow up”, resulting in values which are too large for computer calculations.

The values of α used to detect the convergence of the cost function $J(\Theta)$ are the following:

$$0.005, 0.01, 0.012, 0.014, 0.016, 0.018, 0.1, 0.2, 0.3.$$

The following procedure is carried out for the mathematics course and the Portuguese one:

- implement gradient descent using the function 5;
- compute the mean squared error using k-fold cross validation function;
- repeat this process for all α values;

Listing 5: Gradient Descent function

```
def gradientDescent(X, y, theta, alpha, num_iters):
    """
```

```
3     Take numpy arrays X, y and theta and update theta by taking
          num_iters gradient steps with learning rate alpha
4
5      Return: theta and the list of the cost of theta (J_history)
           during each iteration
6     """
7
8     m=len(y)
9     J_history=[]
10
11    for i in range(num_iters):
12        h = np.dot(X, theta)
13        grad = np.dot(X.transpose(), (h - y))
14        theta= theta - alpha * (1 / m) * grad
15
16        J_history.append(computeCost(X,y,theta))
17    return theta, J_history
```

Once the results obtained using all the parameters have been shown, the same procedure is carried out using only the 10 best parameters (i.e. using the data splitting explained in previous section).

The code used to accomplish this part is the following:

Listing 6: Execution of the model of the math course

```
1 data = pd.read_csv("clean_data/maths.csv")
2 dataM=data
3
4 col_y = 'G3'
5
6 col_drop = ['G1', 'G2'] # All features
7
8 # all
9 X_all, y, X_df = drop_features_get_xy(data, col_drop, col_y)
10
11 # 10 top
12 top_10= ['failures', 'Medu', 'higher', 'age', 'Fedu', 'goout', '
      romantic', 'traveltime', 'Mjob', 'address']
13 X_top10, y, X_df = take_features_get_xy(data, top_10, col_y)
14
```

```
15 X_list = [X_all, X_top10]
16
17 alphaList=[0.005, 0.01, 0.012, 0.014, 0.016, 0.018, 0.1, 0.2, 0.3]
18 iterations = 300
19 for x in X_list:
20     J_history2 = []
21
22     m = x.shape[0]
23
24     x, mean_X, std_X = featureNormalization(x)
25     x = np.append(np.ones((m, 1)), x, axis=1)
26
27     y = y.reshape(m, 1)
28
29     n = x.shape[1]
30
31     y_axis = [*range(iterations)]
32
33     for a in alphaList:
34         theta2 = np.zeros((n,1))
35         theta2, J_history2 = gradientDescent(x, y, theta2, a,
               iterations)
36
37         c = Kfold(x, y, a, 10, iterations)
38         print(f'alpha: {a}, mean_error: {round(c, 4)}')
39
40         plt.plot(y_axis, J_history2, label=a)
41
42     plt.title("Cost function using Gradient Descent for diffrent
          learning rates")
43     plt.xlabel("Iteration")
44     plt.ylabel("$J(\Theta)$")
45     plt.legend()
46     plt.show()
```

Listing 7: Execution of the model of the Portuguese course

```
1 data = pd.read_csv("clean_data/portuguese.csv")
2 dataP=data
3
```

```
col_y = 'G3'

col_drop = ['G1', 'G2'] # All features

# all
X_all, y, X_df = drop_features_get_xy(data, col_drop, col_y)

# 10 top
top_10= ['failures', 'higher', 'school', 'studytime', 'Medu', 'Fedu'
    , 'Dalc', 'Walc', 'reason', 'address']
X_top10, y, X_df = take_features_get_xy(data, top_10, col_y)

X_list = [X_all, X_top10]

alphaList=[0.005, 0.01, 0.012, 0.014, 0.016, 0.0199, 0.1, 0.1001,
    0.2, 0.3]
iterations = 300
for x in X_list:
    J_history2 = []

    m = x.shape[0]

    x, mean_X, std_X = featureNormalization(x)
    x = np.append(np.ones((m, 1)), x, axis=1)

    y = y.reshape(m, 1)

    n = x.shape[1]

    y_axis = [*range(iterations)]

    for a in alphaList:
        theta2 = np.zeros((n,1))
        theta2, J_history2 = gradientDescent(x, y, theta2, a,
            iterations)

        c = Kfold(x, y, a, 10, iterations)
        print(f'alpha: {a}, mean_error: {round(c, 4)}')

```

```
40          plt.plot(y_axis, J_history2, label=a)
41
42      plt.title("Cost function using Gradient Descent for diffrent
            learning rates")
43      plt.xlabel("Iteration")
44      plt.ylabel("$J(\Theta)$")
45      plt.legend()
46      plt.show()
```

5 Results

This section shows the results obtained using the code explained in the previous section. The first part concerns the results obtained using the math course dataset, the second part concerns the results obtained with the Portuguese course dataset.

5.1 Results of regression G3 for the Math data

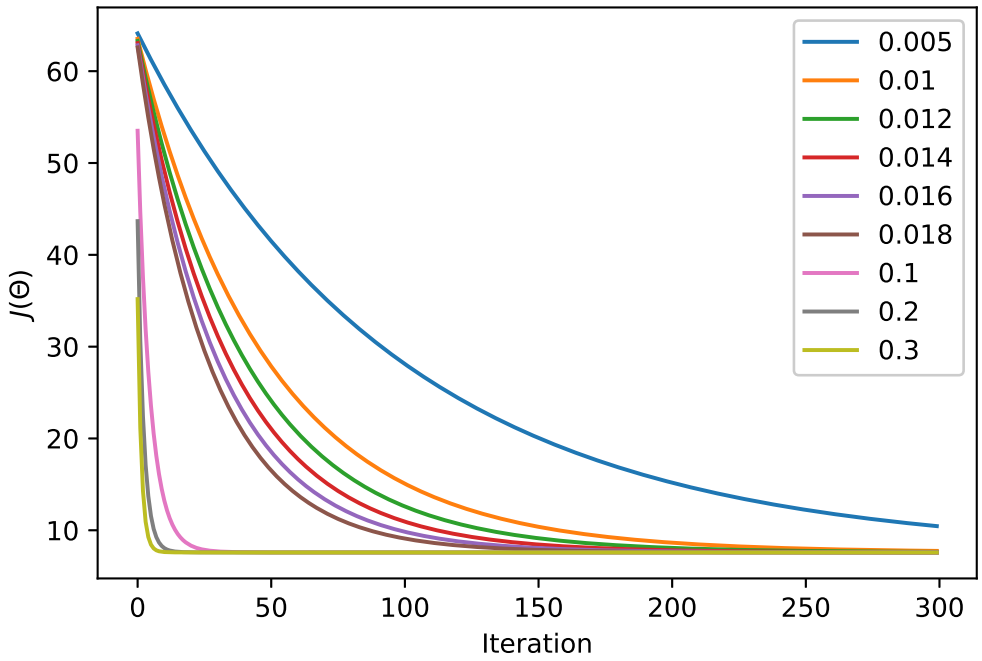


Figure 10: Cost function for different α for data with all features

α	$J(\theta)$
0.005	18.1586
0.01	12.5738
0.012	11.9432
0.014	11.5469
0.016	11.2694
0.018	11.0595
0.1	10.0492
0.2	10.0667
0.3	10.0673

Table 3: Results Math with all features

The best model found (for α = 0.1):

$$h(x) = 10.4 + 20.23 \cdot x_1 + 0.63 \cdot x_2 + -0.48 \cdot x_3 + 0.23 \cdot x_4 + 0.32 \cdot x_5 + -0.1 \cdot x_6 + 0.5 \cdot x_7 + -0.11 \cdot x_8 + -0.17 \cdot x_9 + 0.46 \cdot x_{10} + -1.28 \cdot x_{11} + -0.45 \cdot x_{12} + -0.42 \cdot x_{13} + 0.17 \cdot x_{14} + -0.16 \cdot x_{15} + -0.07 \cdot x_{16} + 0.3 \cdot x_{17} + 0.19 \cdot x_{18} + -0.52 \cdot x_{19} + 0.21 \cdot x_{20} + 0.3 \cdot x_{21} + -0.66 \cdot x_{22} + -0.24 \cdot x_{23} + 0.34 \cdot x_{24} + -0.25 \cdot x_{25} + 0.45 \cdot x_{26} + 0.02 \cdot x_{27} + 0.29 \cdot x_{28} + -0.15 \cdot x_{29} + 0.31 \cdot x_{30} + -0.42 \cdot x_{31} + 0.05 \cdot x_{32} + 0.12 \cdot x_{33} + -0.19 \cdot x_{34} + -0.1 \cdot x_{35} + 0.41 \cdot x_{36} + -0.14 \cdot x_{37} + -0.09 \cdot x_{38} + 0.14 \cdot x_{39} + 0.15 \cdot x_{40} + -0.06 \cdot x_{41} + -0.04 \cdot x_{42}$$

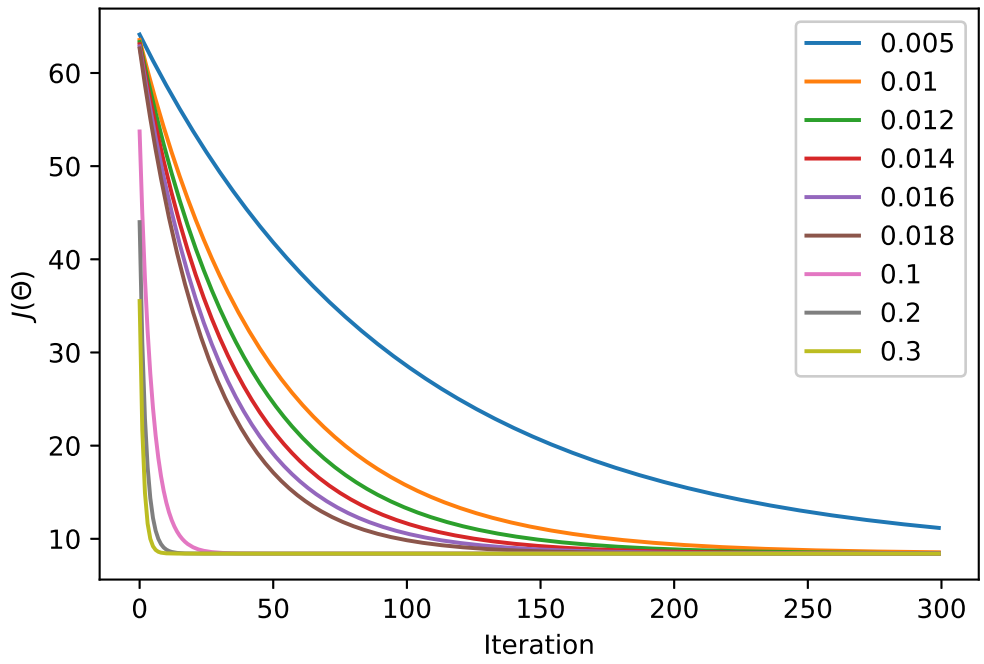


Figure 11: Cost function for different α for data with top 10 features

α	$J(\theta)$
0.005	12.9262
0.01	9.4852
0.012	9.3323
0.014	9.2892
0.016	9.2824
0.018	9.2865
0.1	9.3213
0.2	9.3213
0.3	9.3213

Table 4: Results Math with 10 features

The best model found (for α = 0.0016):

$$h(x) = 10.42 + -1.34 \cdot x_1 + 0.5 \cdot x_2 + 0.23 \cdot x_3 + -0.09 \cdot x_4 + -0.05 \cdot x_5 + -0.49 \cdot x_6 + -0.43 \cdot x_7 + -0.15 \cdot x_8 + 0.2 \cdot x_9 + -0.17 \cdot x_{10} + 0.31 \cdot x_{11} + -0.2 \cdot x_{12} + -0.42 \cdot x_{13}$$

Figure 10 and 11 show the convergence of $J(\Theta)$ using different values of α. It' clear that $\alpha = 0.005$ is too small: the trend is too slow and the cost function remains too large even after 300 iterations. Increasing α the cost function reaches lower values and it converges faster and faster. $\alpha = 0.3$ is the best value among those observed. Increasing α to higher values can be risky inasmuch the graph could blow-up to unexpected values.

5.2 Results of regression G3 for the Portuguese data

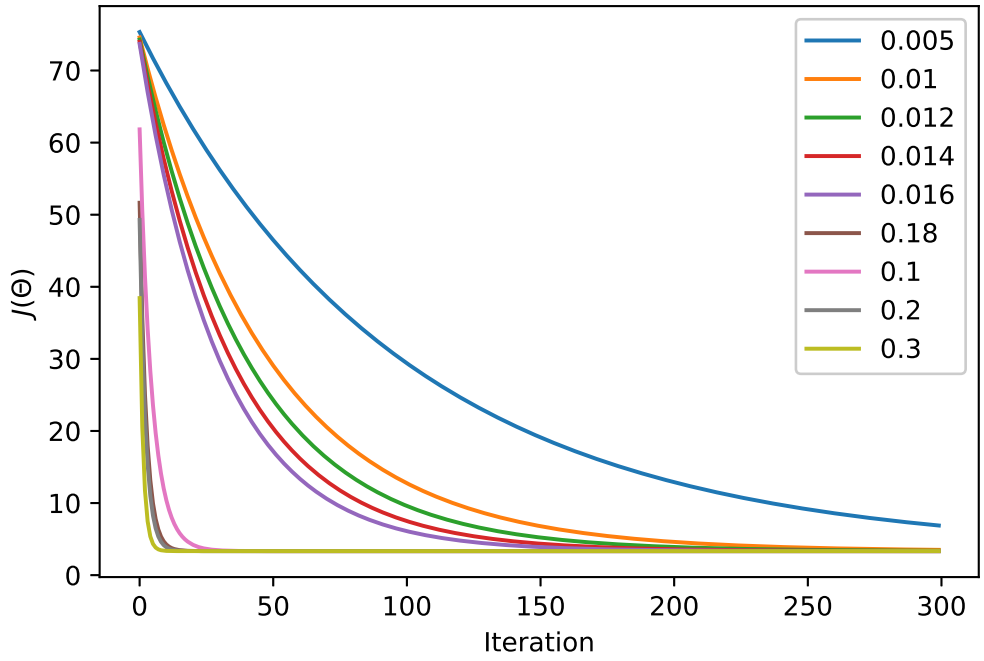


Figure 12: Cost function for different α for data with all features

α	$J(\theta)$
0.005	10.6939
0.01	4.5815
0.012	4.1968
0.014	4.0423
0.016	3.9788
0.018	3.9413
0.1	3.9347
0.2	3.9347
0.3	3.9347

Table 5: Results Portuguese with all features

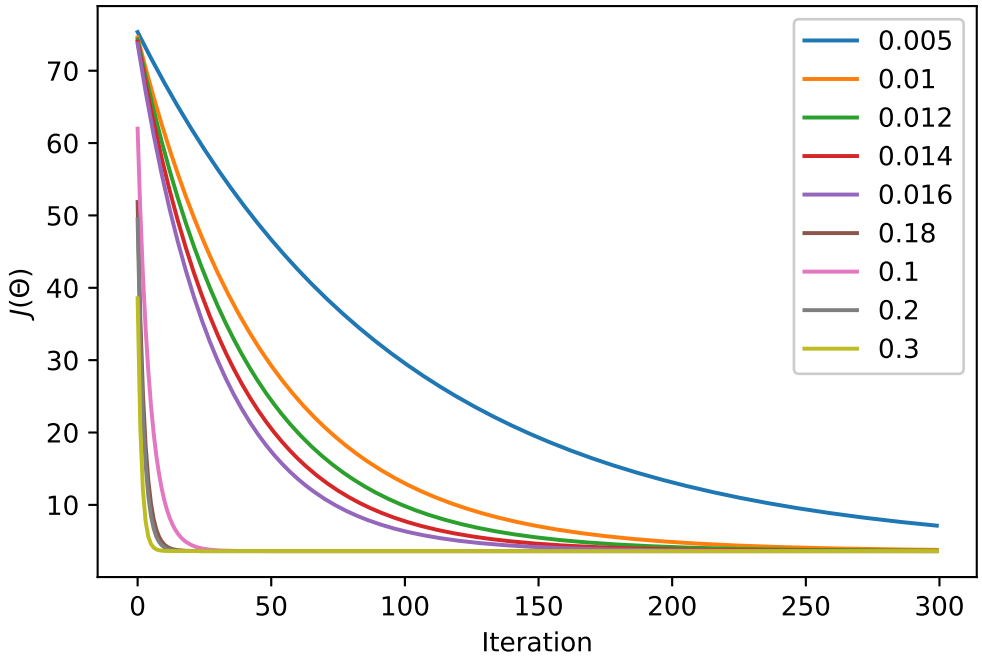


Figure 13: Cost function for different α for data with top 10 features

The best model found (for α = 0.1):

$$h(x) = 11.91 + -0.57 \cdot x_1 + -0.31 \cdot x_2 + 0.19 \cdot x_3 + 0.15 \cdot x_4 + 0.14 \cdot x_5 + 0.06 \cdot x_6 + 0.04 \cdot x_7 + 0.18 \cdot x_8 + 0.05 \cdot x_9 + 0.34 \cdot x_{10} + -0.84 \cdot x_{11} + -0.4 \cdot x_{12} + -0.01 \cdot x_{13} + -0.09 \cdot x_{14} + 0.11 \cdot x_{15} + -0.09 \cdot x_{16} + 0.53 \cdot x_{17} + 0.11 \cdot x_{18} + -0.21 \cdot x_{19} + 0.15 \cdot x_{20} + -0.14 \cdot x_{21} + -0.08 \cdot x_{22} + -0.19 \cdot x_{23} + -0.1 \cdot x_{24} + -0.27 \cdot x_{25} + -0.18 \cdot x_{26} + -0.11 \cdot x_{27} + 0.17 \cdot x_{28} + -0.1 \cdot x_{29} + 0.06 \cdot x_{30} + 0.08 \cdot x_{31} + 0.07 \cdot x_{32} + -0.06 \cdot x_{33} + 0.05 \cdot x_{34} + -0.16 \cdot x_{35} + 0.2 \cdot x_{36} + -0.0 \cdot x_{37} + 0.02 \cdot x_{38} + -0.14 \cdot x_{39} + 0.09 \cdot x_{40} + 0.06 \cdot x_{41} + -0.09 \cdot x_{42}$$

α	$J(\theta)$
0.005	8.9664
0.01	4.2415
0.012	4.0093
0.014	3.9267
0.016	3.8961
0.018	3.8789
0.1	3.8715
0.2	3.8715
0.3	3.8715

Table 6: Results Portuguese with 10 features

Having the same values for alpha, figures 12 and 13 show how the results obtained using the dataset of the Portuguese course are quite similar to those obtained with the dataset of the math course: $\alpha = 0.005$ is too small and the resulting $J(\Theta)$ converges too slow and to a value too high. Increasing α the mean square error converges to a better value.

The best model found (for α = 0.1):

$$h(x) = 11.91 + -0.87 \cdot x_1 + 0.51 \cdot x_2 + -0.46 \cdot x_3 + 0.36 \cdot x_4 + 0.16 \cdot x_5 + 0.17 \cdot x_6 + -0.27 \cdot x_7 + -0.2 \cdot x_8 + 0.16 \cdot x_9 + 0.14 \cdot x_{10} + -0.0 \cdot x_{11} + -0.02 \cdot x_{12} + -0.01 \cdot x_{13}$$

6 Conclusions

The linear regression task for the dataset was successfully completed. For the list of alphas, alphas were found that allows finding the best matches in the least number of iterations.

In both cases, for data from participants in math and Portuguese lessons, models for a reduced number of features give better results (smaller mean squared error).

For the 'math' set with all features, the best result was achieved by the model with $\alpha = 0.1$.

For the 'math' set with 10 features, the model with $\alpha = 0.016$ achieved the best result.

For the 'portuguese' set with all features, the best result was achieved by the model with $\alpha = 0.1$.

For the 'portuguese' set with 10 features, the best result was achieved by the model with $\alpha = 0.1$.

Regression models allow the prediction of the final grade (G3) of students.