

## **Final Project Report**

**By:** Mahmoud Zayad, Parikshit Solunke, Damian Charczuk

### **Problem Selection:**

There are many different factors that influence the performance of a student in their academic endeavors. The best way to determine which of these features are most impactful on their performance is through the use of building regression and clustering models. This way we can see how to help struggling students by looking at variables such as extracurriculars, study time, family relationships, and more. Through the use of the aforementioned models this may provide an insight into the possible solutions that the educational system should implement.

### **Data Collection:**

For our project, we used a dataset provided by the University of California Irvine. The dataset was pulled from their Machine Learning Repository. The dataset we are using contains performance of students in secondary education at two Portuguese schools. The data was collected through questionnaires and reports on students' academic performance. It contains thirty-two features and is based on the students performance in the Portuguese language course. We chose our response variable to be the "G3" column, which is the final grade. The other two Grade variables are G1 and G2 which are earlier grades. All the grades are on a scale of 0-20. It also has 30 other features which include various details about the student ranging from their parents education and economic status to the amount of daily study time. The categorical variables are either binary (eg access to internet: yes/no) or ordinal (Health: 0-5, 0 being the worst and 5 being the healthiest).

### **Data Preparation:**

We encountered no issues in regards to the quality of our dataset. There were no missing values for any of the data entries and most of the features that we chose to use were numeric features. The non-numeric features were converted to integers by creating dummy variables. Once all the feature values were transformed into numeric values, we were able to move to the next step of the data science pipeline.

```
x_dummies=pd.get_dummies(x,drop_first=True)
x_dummies.head()
```

	age	Medu	Fedu	traveltime	studytime	failures	famrel	goout	G1	G2	sex_M	address_U	Pstatus_T	paid_y
0	18	4	4	2	2	0	4	4	5	6	0	1	0	
1	17	1	1	1	2	0	5	3	5	5	0	1	1	
2	15	1	1	1	2	3	4	2	7	8	0	1	1	
3	15	4	2	1	3	0	3	2	15	14	0	1	1	
4	16	3	3	1	2	0	4	2	6	10	0	1	1	

### **Data Exploration:**

Although our dataset contained complete data entries, we decided to drop several irrelevant features to simplify the models we ultimately created to analyze our dataset. These features included the school name, wine/alcohol consumption, reason for going to school, the occupations of the parents, and the relationship the guardians had with the student among other features. We decided to use the final grade (the “G3” column) as the response variable since the final grade is a result of a number of the factors present in our dataset. Our goal was to find which of these features made the largest impact on the final grade.

To find the features that did not correlate with our response variable (“G3”), we grouped the data by the unique values for certain features and looked at the mean of the final grade

("G3") column. Features that had relatively the same mean for each unique value were dropped because this showed that the feature does not hold a strong correlation with our response variable.

```
print(df['reason'].value_counts())
print(df.groupby(by="reason")['G3'].mean())
```

```
course      285
home        149
reputation  143
other        72
Name: reason, dtype: int64
reason
course      11.547368
home        12.181208
other       10.694444
reputation  12.944056
Name: G3, dtype: float64
```

```
print(df.groupby(by="health")['G3'].mean())
```

```
health
1      12.477778
2      12.192308
3      11.838710
4      12.305556
5      11.469880
Name: G3, dtype: float64
```

```
print(df['famsup'].value_counts())
print(df.groupby(by="famsup")['G3'].quantile([0.25,0.5,0.75]))
```

```
yes      398
no       251
Name: famsup, dtype: int64
famsup
no      0.25    10.0
        0.50    12.0
        0.75    14.0
yes     0.25    10.0
        0.50    12.0
        0.75    14.0
Name: G3, dtype: float64
```

## Data Modeling:

### Regression:

The regression models we built started by using three different methods of regression in order to see which of them is most effective with the dataset. The methods we went through were Linear, Ridge, and Lasso regression. We used all the features for the comparison of these different methods. Ridge regression gave the best score and Lasso the worst. The Lasso regression model ignored all the features except “G2.” Linear regression shared similar results to the Ridge model but it was slightly worse, the ridge model had an adjusted r-squared of  $\sim 0.8214$ .

Thus we only used Ridge regression when building the rest of the models. We started creating subsets of features to use in various models. We first tested the use of only two features, “G1” and “G2,” intuition leads us to believe that earlier grades would be the greatest predictor for “G3.” With that subset we had an adjusted r-squared of 0.874. From here on we continued to use the two aforementioned features in the following subsets.

`r2: 0.8772557652626836 Adjusted r2 0.8740256538222279`


We then tested the first half of the features within the dataframe and the second half. The first half contained features age: Medu, Fedu, traveltime, studytime, failures, famrel, goout, and G1. The features included parental education, family relations, how the student has performed in the past and how they spend their time i.e. studying. These features were not strong indicators in predicting G3. The second half included: G2, sex\_M, address\_U, Pstatus\_T, paid\_yes, nursery\_yes, higher\_yes, internet\_yes, and romantic\_yes. Comparing these subsets the second half performed much better by  $\sim 0.15$ . The adjusted r-squared of the second half was  $\sim 0.852$  while the first half subset had a value of  $\sim 0.709$ . This tells us that it is possible that features related to age and family variables do not have a significant impact on G3.

From here on we began testing various subsets that included G1, and G2 since they had that model using those features as the highest performing one. After comparing these various subsets we found that the combination of different features with G1 and G2 had little impact on the prediction. The best subset being G1, G2, studytime, and failures w/ an adjusted r-squared of  $\sim 0.877$ . Other subsets would lower the value compared to the model just using G1 and G2 or increase it slightly. The lowest value was  $\sim 0.871$  with the same subset as the previous one with the age feature included. This shows that previous performance and study time are the greatest predictors of how a student will perform.

### Clustering:

We started our clustering process by scaling the features using StandardScaler and choosing G1 (first period grade), G2 (second period grade), address (urban/rural), parent status (“Pstatus”), and whether the students wants to pursue higher education (“higher”) as the selected features. Then, we ran K-Means models on the selected features while incrementing the number of clusters from 3 to 9 in the process (as seen in the picture below) to find the number of clusters that result in the best-performing model. By using the silhouette score, we were able to conclude that there are 6 clusters for this subset of features.

```

>  M4
#We will use the silhouette score to determine the appropriate number of clusters
x_subset=x_clustering[:,[8,9,11,12,15]]
k_range=range(3,9)
for k in k_range:
    clustering = KMeans(n_clusters = k, init = 'k-means++', n_init = 10).fit(x_subset)
    clusters=clustering.labels_
    print(metrics.silhouette_score(x_subset, clusters, metric = "euclidean"),"k:",k)
0.3114492827810512 k: 3
0.3783637711278405 k: 4
0.4526937564281484 k: 5
0.46779947442163183 k: 6
0.44969332352354957 k: 7
0.45707074411590204 k: 8

```

We then used the following hyperparameters of: 6 n\_clusters, k-means++ initialization and 10 iterations to create a K-Means clustering model.

To visualize the resulting clusters, we used the Seaborn sns.stripplot() function to plot the resulting clusters using G1 and G2 respectively.



We then used `AgglomerativeClustering()` with the same subset of features and the following hyperparameters of: 6 clusters and using ward linkage, to create another clustering model. We then used the `sns.stripplot()` function to plot the clusters as before.



Using this feature set and the resulting plots, we saw that from the 6 identical clusters, two are relatively high-scoring, two low-scoring, one that is relatively average, and one that has a lot of variation and includes really high scoring students as well as low scoring students. Additionally, from the plots above, we concluded that K-Means and agglomerative clustering have similar results for our dataset. As a result, we decided to use K-Means moving forward. We decide to not use DBSCAN at all because it performs poorly when there is a large number of features.

Next, we tried using a different set of features. This time, we used “famrel” (which puts the quality of family relationships on a scale from 1 to 5) and “paid” (whether or not the student paid for extra classes). As before, we tried different combinations of parameters and used the

silhouette score to determine the appropriate number of clusters. We concluded that this subset of features can be clustered into 8 clusters.

```

> M4

#We will use the silhouette score to determine the appropriate number of clusters
x_subset=x_clustering[:,[5,13]]
k_range=range(3,10)
for k in k_range:
    clustering = KMeans(n_clusters = k, init = 'k-means++', n_init = 10).fit(x_subset)
    clusters=clustering.labels_
    print(metrics.silhouette_score(x_subset, clusters, metric = "euclidean"),"k:",k)
0.9234412933670707 k: 3
0.9670461833718402 k: 4
0.9736083036260131 k: 5
0.9895837911241617 k: 6
0.99768875192604 k: 7
0.9984591679506933 k: 8
0.9984591679506933 k: 9

```

Then, we created the K-Means model using 8 as the number of clusters, k-means++ initialization and 10 iterations.

As we did with the previous models, we plotted the resulting clusters and got the following results:







While the silhouette score for the clustering is perfect, there was not a clear pattern for a cluster in terms of the G1 and G2 scores. So in the next subset of features, we used “address\_U”, “Pstatus\_T”, “paid\_yes”, “nursery\_yes”, and “internet\_yes”. Using the same method as before, we found that 10 clusters gives us the highest silhouette score so we built a K-Means model using 10 as the hyperparameter for the number of clusters.

```

#We will use the silhouette score to determine the appropriate number of clusters
x_subset=x_clustering[:,[11,12,13,14,16]]
k_range=range(3,11)
for k in k_range:
    clustering = KMeans(n_clusters = k, init = 'k-means++', n_init = 10).fit(x_subset)
    clusters=clustering.labels_
    print(metrics.silhouette_score(x_subset, clusters, metric = "euclidean"),"k:",k)
0.4767507751240303 k: 3
0.5126391383509965 k: 4
0.6128667699327223 k: 5
0.7290964762842583 k: 6
0.7894390172032485 k: 7
0.8368623813994333 k: 8
0.8616489561115708 k: 9
0.8975422496454084 k: 10

```



```

> MI
#kmeans
clustering = KMeans(n_clusters = 10, init = 'k-means++', n_init = 10).fit(x_subset)
clusters=clustering.labels_
df2=x
df2["clusters"] = pd.Series(clusters)

```



Based on these plots, we concluded that while the silhouette score for the clustering is good, there is not a clear pattern for a cluster in terms of the G1 and G2 scores and these are virtually equally distributed across clusters. So for our last clustering model, we used “G1” and “G2” as our input features and found that the feature set can be clustered into 3 clusters so we built the model accordingly.

```

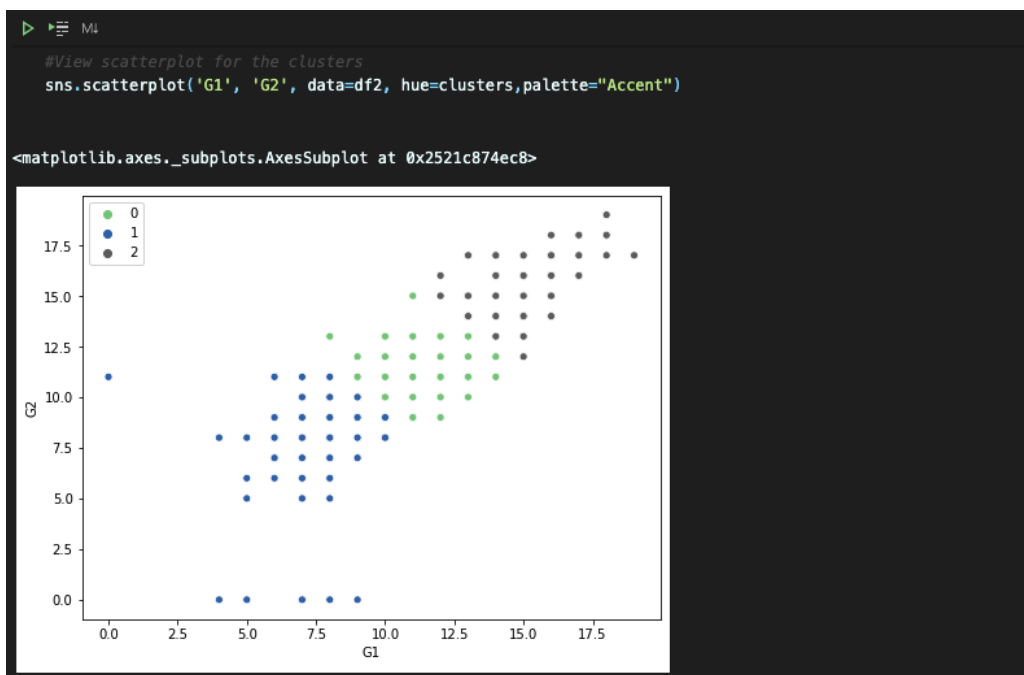
▶ M4

#We will use the silhouette score to determine the appropriate number of clusters
x_subset=x_clustering[:,[8,9]]
k_range=range(3,11)
for k in k_range:
    clustering = KMeans(n_clusters = k, init = 'k-means++', n_init = 10).fit(x_subset)
    clusters=clustering.labels_
    print(metrics.silhouette_score(x_subset, clusters, metric = "euclidean"),"k:",k)

0.44553968283059003 k: 3
0.42926131521960736 k: 4
0.4451936834575136 k: 5
0.4267317343204977 k: 6
0.39064889527413793 k: 7
0.38625946646208004 k: 8
0.40164807938918234 k: 9
0.39530109457456786 k: 10

```

The K-Means model using 3 as the number of clusters, k-means++ initialization and 10 iterations results in the following plot, in which we can clearly see the clusters aligning from lower scores for both G1 and G2 to higher scores, with little to no overlap. We can also conclude that usually a low G1 score correlates with a low G2 score.



**Results:**

Through our regression models, we found that the best predictors of the final grade were the earlier grades of the grading period, the amount of time the student spent studying, and the number of past class failures. The adjusted R<sup>2</sup> for that model was 0.875.

Through our clustering models, we found that G1 and G2 are the best features to use to create accurate clusters. However, our findings offer no valuable insight since it is rather obvious that the grades the student received earlier in the grading period have a large effect on the final term grade. Perhaps if the dataset contained more than a couple hundred data points, we could have received more significant findings.