Linear Regression: Math scores predictions based on different traits

David Parro

August 2021

1 Data description and objective

I have chosen a data-set with 7 features and a target column, math score. They range from gender to all the different grades that they have got in their three classes.
The objective is to predict as accurately as possible the math score knowing all the other features.
Code is added after the report

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	reading_score	writing_score	math_score
0	female	group B	bachelor's degree	standard	none	72	74	72
1	female	group C	some college	standard	completed	90	88	69
2	female	group B	master's degree	standard	none	95	93	90
3	male	group A	associate's degree	free/reduced	none	57	44	47
4	male	group C	some college	standard	none	78	75	76

It is important to know about our features types, using df.types()

gender	object
parental_level_of_education	object
lunch	object
test_preparation_course	object
reading_score	int64
writing_score	int64
math_score	int64

2 Feature engineering

The actions taken can be briefly summarized into checking whether there are any empty entries and decide what to to do.

-If any categorical feature is missing it will be substituted by the most repeated value.
-If any numerical feature is missing it will be substituted by the average of the other values
-Finally if more than half of the features are missing that observation will be dropped

The result of this procedure is:

gender_male	race_ethnicity_group B	race_ethnicity_group C	race_ethnicity_group D	race_ethnicity_group E	parental_level_of_education_bachelor's degree	parenta
0	1	0	0	0	1	
0	0	1	0	0	0	
0	1	0	0	0	0	
1	0	0	0	0	0	
1	0	1	0	0	0	

3 Linear Regression models

On this section I will approach three different linear regression models:

- Standard linear regression
- LASSO linear regression
- Ridge linear regression

Using cross validation with 3-fold. In order to do this we created a k-fold object, which will be used al-through the code.

3.1 Standard linear regression

Standard transformation does not change standard LR, so it will not be included in our pipeline. I will compare the model with the 3 folds and decide which is the more "important" feature in order to decide the math score of new data. The results are as follow:

Feature	Importance
parental_level_of_education_master's degree	-0.258459
race_ethnicity_group C	0.021839
parental_level_of_education_bachelor's degree	0.200683
race_ethnicity_group B	0.298907
parental_level_of_education_some college	0.346957
race_ethnicity_group D	0.348056
parental_level_of_education_high school	0.421703
parental_level_of_education_some high school	0.484809
lunch_standard	1.426520
test_preparation_course_none	1.476546
race_ethnicity_group E	1.917065
reading_score	4.026635
gender_male	6.613170
writing_score	10.966845

Feature	Importane
parental_level_of_education_master's degree	-0.525221
parental_level_of_education_bachelor's degree	-0.291586
parental_level_of_education_some high school	-0.123599
race_ethnicity_group C	-0.119797
parental_level_of_education_high school	0.045315
parental_level_of_education_some college	0.415900
race_ethnicity_group D	0.418690
race_ethnicity_group B	0.751775
lunch_standard	1.317731
race_ethnicity_group E	1.621866
test_preparation_course_none	2.097788
reading_score	4.567896
gender_male	6.514808
writing_score	9.363337

Feature	Importance
parental_level_of_education_master's degree	-0.773652
race_ethnicity_group C	-0.612283
parental_level_of_education_bachelor's degree	-0.552462
parental_level_of_education_some college	-0.163441
race_ethnicity_group D	-0.125605
race_ethnicity_group B	0.005950
parental_level_of_education_some high school	0.198798
parental_level_of_education_high school	0.291969
race_ethnicity_group E	1.015074
lunch_standard	1.454153
test_preparation_course_none	1.698867
reading_score	3.536557
gender_male	6.167246
writing_score	11.393368

3.1.1 Conclusion

After working with the three folds we can conclude that the most important feature for this standard LR model is the writing score, the gender and finally reading score. We can conclude that most of these features are not natural thus race is not very deterministic for math scores, even so parental education is the least important feature.

3.2 Ridge Regression

3.2.1 Hyper parameter tuning: manual vs grid

Studying from:

- ridge_alphas = np.geomspace(1e-9, 1e0, num=10) and polynomial features of degree 3

I can conclude that our best $\alpha = 1$ with a R_2 score of 0.7811531136397738
But if we use grid, that does this task automatically and better we can conclude, adding it to our pipeline:

- R_2 score = 0.8641092594997569
- Polynomial_features: 1
- α: 4.0

Which is without doubt a better and simpler model.

3.2.2 Conclusion

I conclude that, given enough time, letting the grid function test for different degrees, comparing the mean squared error of the test set and the training set. It is also important to use the R_2 score as a decision variable, in order to decide the best model.

Feature	Importance
lunch_standard	-1.383785
parental_level_of_education_some college	-0.281988
parental_level_of_education_bachelor's degree	-0.184247
parental_level_of_education_some high school	-0.082219
reading_score	0.000000
race_ethnicity_group E	0.027454
race_ethnicity_group D	0.133122
race_ethnicity_group C	0.191074
test_preparation_course_none	0.223934
parental_level_of_education_high school	0.438403
parental_level_of_education_master's degree	1.055289
race_ethnicity_group B	1.222822
writing_score	1.988680
gender_male	4.298263

In this model we conclude that the most important feature in order to decide the math score is the gender of the student.

3.3 Lasso regression

3.3.1 Hyper parameter tuning

Studying from:

- alphas = np.geomspace(1e-9, 1e0, num=10) and polynomial features of degree 3

I can conclude that our best $\alpha = 0.1$ with a R_2 score of 0.8590101013738956
Once we have set up our model, it is time to analyse the features!

3.3.2 Conclusions

From this feature table:

0	1
parental_level_of_education_some college	-0.137959
reading_score	0.000000
race_ethnicity_group C	-0.000000
race_ethnicity_group D	-0.000000
race_ethnicity_group E	0.000000
parental_level_of_education_bachelor's degree	0.000000
parental_level_of_education_high school	-0.000000
parental_level_of_education_some high school	-0.000000
lunch_standard	-0.000000
parental_level_of_education_master's degree	0.089780
test_preparation_course_none	0.129093
writing_score	1.830519
race_ethnicity_group B	4.769109
gender_male	10.386407

I can conclude that once again gender is the most impactful feature is the gender of the student. The number of 0's is bigger than in the other models as LASSO tends to force more coefficients to 0.

3.4 Which model is the best one?

Our Ridge model obtained by grid gets both the best R_2 score and the simplest model (degree = 1). It would be the best one in order to draw quicker conclusions, given enough time and a better sampling of alphas and degrees may output a better model than one obtained, but it is without doubt a really good model for its simplicity.

3.5 Insights and next steps

The key insight that we can gather from this report is that gender and race, which are natural traits, are for most of the models the most important features.
It is also important to note that writing score seems to be the most correlated test to the math score and that reading score does not seem to have any impact on deciding wether a students succeeds or not at maths.

The data is normally distributed so the mean of the sample is a good estimator of the score of any given student.

3.5.1 Next steps

I would try to explore with more complex models and trying to fit a bigger sample in order to have better representative data.

8 features are not many, I would try to add some new features to the data-sets, such as the students reporting how many hours they studied for a given test (reading, writing or math)

I think that that may be a better estimator overall for this kind of research.

Predicting math's test scores based on individual traits

In [1]:

```
import numpy as np
import numpy.ma as ma
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

from scipy.stats.mstats import normaltest # D'Agostino K^2 Test
from scipy.stats import boxcox

import matplotlib.pyplot as plt
%matplotlib inline
from helper import (plot_exponential_data,
                    plot_square_normal_data)

from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import (StandardScaler,
                                   PolynomialFeatures)
from sklearn.metrics import mean_squared_error

import warnings
warnings.simplefilter("ignore")
```

In [2]:

```
file_path = "data/StudentsPerformance.csv"
df = pd.read_csv(file_path)
```

In [3]:

```
df.shape
```

Out[3]:

```
(1000, 8)
```

In [4]:

```
#I did not have enough memory  to work with the 1000 observations, so I dropped half.
dropped = list()
for i in range(500):
    dropped.append(999-i)
```

In [5]:

```
#Intenta añadir la recta de regresión
```

In [6]:

```
df = df.drop(dropped)
```

In [7]:

```
df.shape
```

Out[7]:

```
(500, 8)
```

1.Data analysis

We are presented a data sete containing 7 features and a target column, math_score, we will reorder the dataframe in order to work in a more standard way.

My goal is to predict as accurately as possible the math score of a certain student based on their gender, race, other scores ,etc...

In [8]:

```
df.head()
```

Out[8]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

In [9]:

```
df.rename(columns={df.columns[1]:"race_ethnicity" ,df.columns[2]:"parental_level_of_edu
                   df.columns[5]: "math_score",df.columns[4]: "test_preparation_course'
                   df.columns[6]: "reading_score",df.columns[7]: "writing_score"}, erro
```

In [10]:

```
df = df[["gender", "race_ethnicity" ,"parental_level_of_education", "lunch",
       "test_preparation_course", "reading_score", "writing_score","math_score"]]
```

In [11]:

```
df.head()
```

Out[11]:

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	rea
0	female	group B	bachelor's degree	standard	none	
1	female	group C	some college	standard	completed	
2	female	group B	master's degree	standard	none	
3	male	group A	associate's degree	free/reduced	none	
4	male	group C	some college	standard	none	

Right now everything seems as a standard dataframe

In [12]:

```
print(df.dtypes)
```

```
gender                         object
race_ethnicity                 object
parental_level_of_education    object
lunch                          object
test_preparation_course        object
reading_score                   int64
writing_score                   int64
math_score                      int64
dtype: object
```

In [13]:

```
df.math_score.mean()
```

Out[13]:

```
65.714
```

Five of our features are strings, in order to work with categorycal features, one hot encoding is needed

2.Feature engineering

2.1 Empty entries

In [14]:

```
1 df.isnull().sum()
```

Out[14]:

```
gender                         0
race_ethnicity                 0
parental_level_of_education    0
lunch                          0
test_preparation_course        0
reading_score                  0
writing_score                  0
math_score                     0
dtype: int64
```

We do not have any empty entries, if we had them we would substitute them by the average value in case of numerical features, and by the most repeated value in case of categoical features.

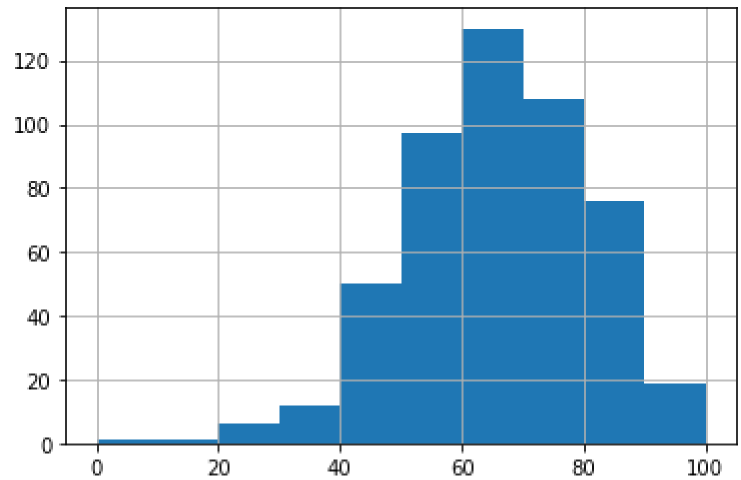
2.2 Is our dataset normally distributed?

In [15]:

```
1 df.math_score.hist()
```

Out[15]:

```
<AxesSubplot:>
```



In [16]:

```
1 normaltest(df.math_score.values)
```

Out[16]:

```
NormaltestResult(statistic=13.704221358790235, pvalue=0.0010572218798674245)
```

We use box cox transformation in order to get to the desired p-value (0.05)

In [17]:

```
#Replacing 0 values by 1 (int) doesnt skew the data too much as there is only one 0.

count = 0
for score in df.math_score:
    if score == 0:
        df.at[count,"math_score"] = 1
    count = count + 1
df.math_score[59]
```

Out[17]:

1

In [18]:

```
bc_result = boxcox(df.math_score.values)
boxcox_math = bc_result[0]
lam = bc_result[1]
lam
```

Out[18]:

1.3653906458825082

In [19]:

```
normaltest(boxcox_math)
```

Out[19]:

NormaltestResult(statistic=0.3115540260638914, pvalue=0.8557499901930166)

p-value > 0.05 thus we cannot reject H0, our data is normally distributed

2.3 One hot encoding

In [20]:

```
one_hot_encode_cols = df.dtypes[df.dtypes == np.object]
one_hot_encode_cols = one_hot_encode_cols.index.tolist()
```

In [21]:

```
df = pd.get_dummies(df, columns=one_hot_encode_cols, drop_first=True)
```

In [22]:

```
for column in df:
    print(column)
```

```
reading_score
writing_score
math_score
gender_male
race_ethnicity_group B
race_ethnicity_group C
race_ethnicity_group D
race_ethnicity_group E
parental_level_of_education_bachelor's degree
parental_level_of_education_high school
parental_level_of_education_master's degree
parental_level_of_education_some college
parental_level_of_education_some high school
lunch_standard
test_preparation_course_none
```

In [64]:

```
df.head()[["gender_male","race_ethnicity_group B","race_ethnicity_group C","parental_le
```

Out[64]:

	gender_male	race_ethnicity_group B	race_ethnicity_group C	parental_level_of_education_bachelor degre
0	0	1	0	
1	0	0	1	
2	0	1	0	
3	1	0	0	
4	1	0	1	

3. Regression models

In [26]:

```
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.model_selection import KFold, cross_val_predict
from sklearn.pipeline import Pipeline
```

We will use K-fold division for better results

In [27]:

```
kf = KFold(shuffle=True, random_state=72018, n_splits=3)
```

3.1 Standard linear regression

We learnt that standard scaling is not needed in order to perform standard linear regression (it does not change the R2 score).

Lets check how it works

Setting the model

In [28]:

```
scores = []

lr = LinearRegression()
s = StandardScaler()

X = df.drop("math_score", axis=1)
y = df.math_score

for train_index, test_index in kf.split(X):
    X_train, X_test, y_train, y_test = (X.iloc[train_index, :],
                                        X.iloc[test_index, :],
                                        y[train_index],
                                        y[test_index])

    X_train_s = s.fit_transform(X_train)

    lr.fit(X_train_s, y_train)

    display(pd.DataFrame(zip(X.columns, lr.coef_)).sort_values(by=1))

    X_test_s = s.transform(X_test)

    y_pred = lr.predict(X_test_s)

    score = r2_score(y_test.values, y_pred)

    scores.append(score)
```

	0	1
9	parental_level_of_education_master's degree	-0.258459
4	race_ethnicity_group C	0.021839
7	parental_level_of_education_bachelor's degree	0.200683
3	race_ethnicity_group B	0.298907
10	parental_level_of_education_some college	0.346957
5	race_ethnicity_group D	0.348056
8	parental_level_of_education_high school	0.421703
11	parental_level_of_education_some high school	0.484809
12	lunch_standard	1.426520
13	test_preparation_course_none	1.476546
6	race_ethnicity_group E	1.917065
0	reading_score	4.026635
2	gender_male	6.613170
1	writing_score	10.966845

	0	1
9	parental_level_of_education_master's degree	-0.525221
7	parental_level_of_education_bachelor's degree	-0.291586

	0	1
11	parental_level_of_education_some high school	-0.123599
4	race_ethnicity_group C	-0.119797
8	parental_level_of_education_high school	0.045315
10	parental_level_of_education_some college	0.415900
5	race_ethnicity_group D	0.418690
3	race_ethnicity_group B	0.751775
12	lunch_standard	1.317731
6	race_ethnicity_group E	1.621866
13	test_preparation_course_none	2.097788
0	reading_score	4.567896

	0	1
9	parental_level_of_education_master's degree	-0.773652
4	race_ethnicity_group C	-0.612283
7	parental_level_of_education_bachelor's degree	-0.552462
10	parental_level_of_education_some college	-0.163441
5	race_ethnicity_group D	-0.125605
3	race_ethnicity_group B	0.005950
11	parental_level_of_education_some high school	0.198798
8	parental_level_of_education_high school	0.291969
6	race_ethnicity_group E	1.015074
12	lunch_standard	1.454153
13	test_preparation_course_none	1.698867
0	reading_score	3.536557
2	gender_male	6.167246
1	writing_score	11.393368

In [29]:

```
scores
```

Out[29]:

```
[0.8369460057628269, 0.9014676096900432, 0.8521539872141326]
```

R2 Score

In [30]:

```
r2 = r2_score(y_pred, y_test)
r2
```

Out[30]:

```
0.8303889713068369
```

3.2 Ridge Regression

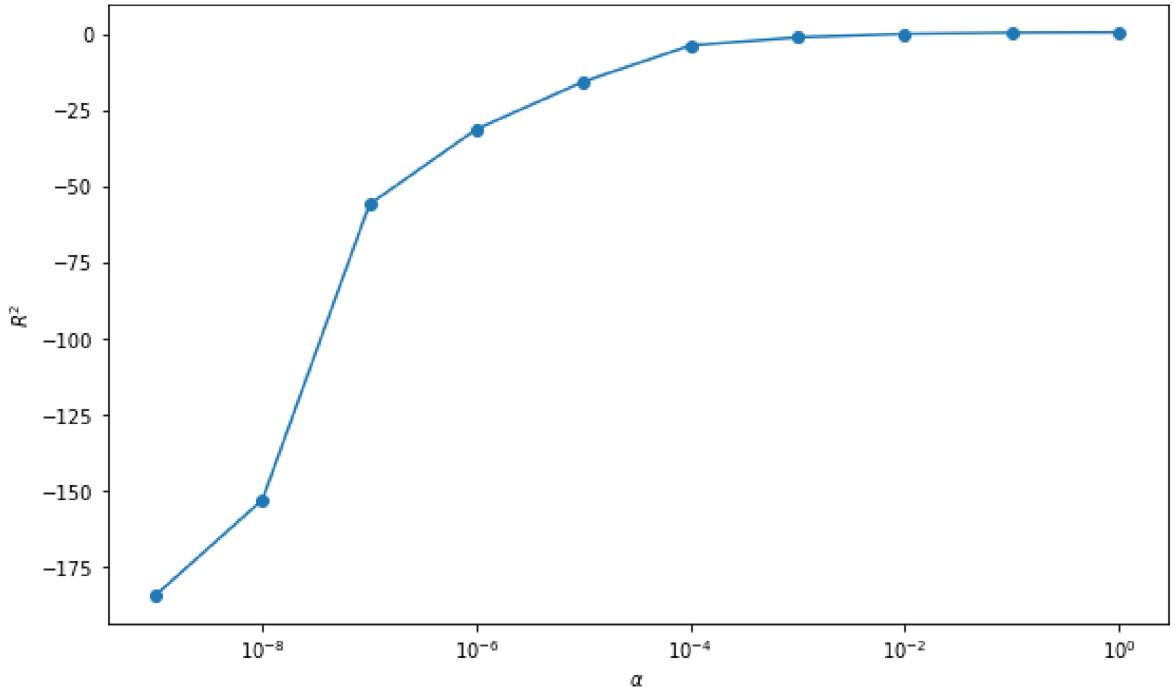
Hyperparameter tuning: Grid vs "Manual" tuning

In [31]:

```
ridge_alphas = np.geomspace(1e-9, 1e0, num=10)

pf = PolynomialFeatures(degree = 3)
ridge_scores = []
ridge_coefs = []


for alpha in ridge_alphas:
    rid = Ridge(alpha=alpha, max_iter=100000)

    ridge_estimator = Pipeline([
        ("poly features", pf),
        ("scaler", s),
        ("Ridge_regression", rid)])

    ridge_predictions = cross_val_predict(ridge_estimator, X, y, cv = kf)

    ridge_score = r2_score(y, ridge_predictions)

    ridge_scores.append(ridge_score)
```

In [32]:

```
plt.figure(figsize=(10,6))
plt.semilogx(ridge_alphas, ridge_scores, '-o')
plt.xlabel('$\\alpha$')
plt.ylabel('$R^2$');
```



In [33]:

```
score_list = list(zip(ridge_alphas,ridge_scores))
len(score_list)
score_list
```

Out[33]:

```
[(1e-09, -184.21612815040905),
 (1e-08, -152.82742141764894),
 (1e-07, -55.77019290631202),
 (1e-06, -31.062383428946276),
 (1e-05, -15.434447628329522),
 (0.0001, -3.5777904964946527),
 (0.001, -0.9252430999887618),
 (0.01, 0.22591093124632633),
 (0.1, 0.6269110660729362),
 (1.0, 0.7811531136397738)]
```

In [34]:

```
ridge_alpha = 0
len(score_list)
for i in range (len(score_list)):
    if score_list[i][1] ==max(ridge_scores):
        print(score_list[i][1])
        ridge_alpha = score_list[i][0]
```

```
0.7811531136397738
```

In [35]:

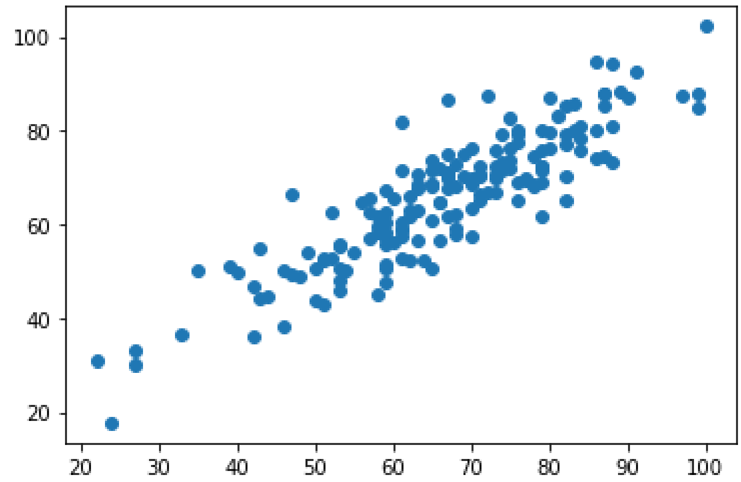
```
#Initialize objects
s = StandardScaler()
lr = LinearRegression()
pf = PolynomialFeatures(degree = 3)
rid = Ridge(alpha=ridge_alpha, max_iter=100000)
ridge_best_estimator = Pipeline([
        ("poly features", pf),
        ("scaler", s),
        ("Ridge_regression", rid)])
```

In [36]:

```
ridge_best_estimator.fit(X_train, y_train)
ridge_best_estimator.predict(X_test)
plt.scatter(y_test, ridge_best_estimator.predict(X_test))
```

Out[36]:

```
<matplotlib.collections.PathCollection at 0x244a1aff6a0>
```



In [37]:

```
ridge_best_predictions = cross_val_predict(ridge_best_estimator, X, y, cv=kf)
```

In [38]:

```
r2_score(y, ridge_best_predictions)
```

Out[38]:

```
0.7811531136397738
```

Using Grid

In [39]:

```
from sklearn.model_selection import GridSearchCV
```

In [40]:

```
np.geomspace(1, 20, 30)
```

Out[40]:

```
array([ 1.        ,  1.10882524,  1.22949341,  1.36329333,  1.51165405,
        1.67616017,  1.8585687 ,  2.06082789,  2.28509798,  2.53377432,
        2.80951292,  3.11525883,  3.45427763,  3.83019022,  4.24701159,
        4.70919365,  5.22167278,  5.78992257,  6.42001229,  7.11867167,
        7.89336283,  8.75235993,  9.70483761, 10.76096889, 11.93203392,
       13.23054038, 14.67035711, 16.26686225, 18.03710745, 20.        ])
```

In [41]:

```
from sklearn.model_selection import GridSearchCV

# Same estimator as before
ridge_estimator = Pipeline([("scaler", StandardScaler()),
        ("polynomial_features", PolynomialFeatures()),
        ("ridge_regression", Ridge())])

params = {
    'polynomial_features__degree': [1, 2, 3],
    'ridge_regression__alpha': np.geomspace(4, 20, 30)
}

grid = GridSearchCV(ridge_estimator, params, cv=kf)
```

In [42]:

```
grid.fit(X, y)
```

Out[42]:

```
GridSearchCV(cv=KFold(n_splits=3, random_state=72018, shuffle=True),
             estimator=Pipeline(steps=[('scaler', StandardScaler()),
                                       ('polynomial_features',
                                        PolynomialFeatures()),
                                       ('ridge_regression', Ridge())]),
             param_grid={'polynomial_features__degree': [1, 2, 3],
                         'ridge_regression__alpha': array([ 4.        ,  4.2
2826702,  4.46956049,  4.7246238 ,  4.99424274,
        5.27924796,  5.58051751,  5.89897953,  6.23561514,  6.59146146,
        6.96761476,  7.36523392,  7.78554391,  8.22983963,  8.69948987,
        9.19594151,  9.72072404, 10.27545421, 10.86184103, 11.48169104,
       12.13691388, 12.82952815, 13.56166768, 14.33558803, 15.15367351,
       16.01844446, 16.93256509, 17.89885162, 18.92028098, 20.        ])})
```

In [43]:

```
grid.best_score_, grid.best_params_
```

Out[43]:

```
(0.8641092594997569,
 {'polynomial_features__degree': 1, 'ridge_regression__alpha': 4.0})
```

In [44]:

```
y_predict = grid.predict(X)
```

In [45]:

```
# This includes both in-sample and out-of-sample
r2_score(y, y_predict)
```

Out[45]:

```
0.8781502005670764
```

In [46]:

```
grid.best_estimator_.named_steps['ridge_regression'].coef_
```

Out[46]:

```
array([ 0.        ,  4.83522078,  9.64621561,  6.28953901,  0.31123453,
       -0.2462869 ,  0.23545716,  1.47871442, -0.16804017,  0.22044023,
       -0.48972383,  0.188726  ,  0.1216128 ,  1.47429793,  1.61915634])
```

In [60]:

```
pd.DataFrame(zip(X.columns, rid.coef_)).sort_values(by=1)
```

	0	1
12	lunch_standard	-1.383785
10	parental_level_of_education_some college	-0.281988
7	parental_level_of_education_bachelor's degree	-0.184247
11	parental_level_of_education_some high school	-0.082219
0	reading_score	0.000000
6	race_ethnicity_group E	0.027454
5	race_ethnicity_group D	0.133122
4	race_ethnicity_group C	0.191074
13	test_preparation_course_none	0.223934
8	parental_level_of_education_high school	0.438403
9	parental_level_of_education_master's degree	1.055289

3.3 LASSO Regression

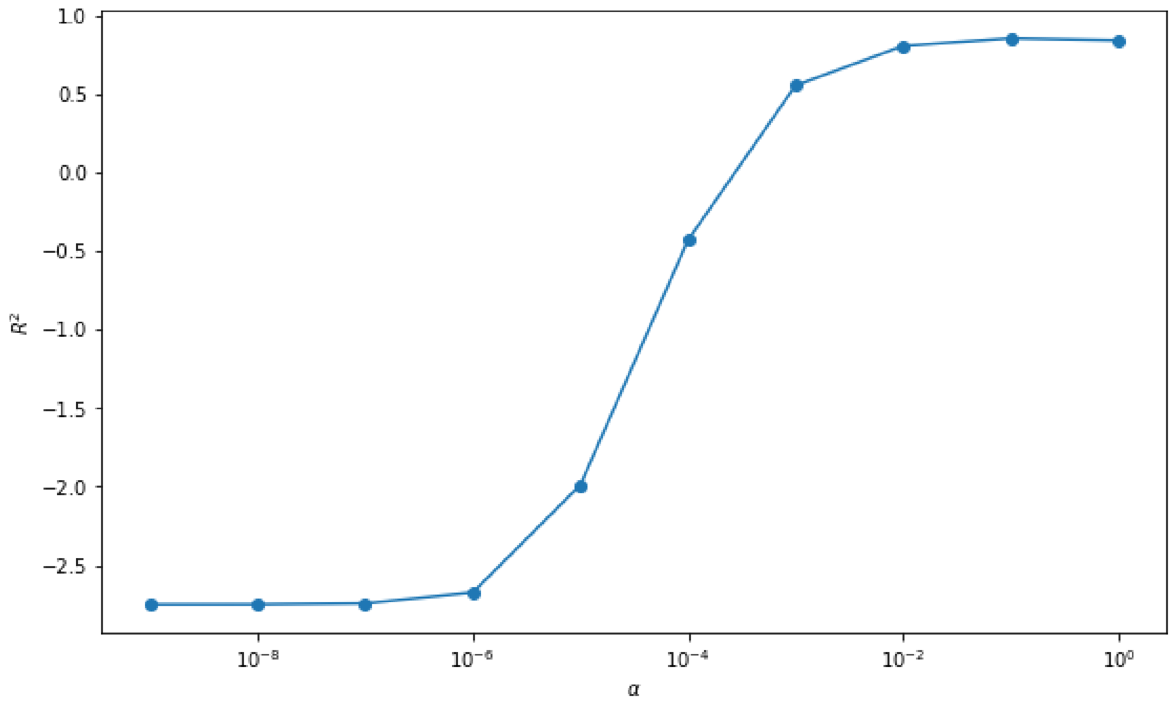
Hyperparameter tuning

In [48]:

```
alphas = np.geomspace(1e-9, 1e0, num=10)

pf = PolynomialFeatures(degree = 3)
scores = []
coefs = []


for alpha in alphas:
    las = Lasso(alpha=alpha, max_iter=100000)

    estimator = Pipeline([
        ("poly features", pf),
        ("scaler", s),
        ("lasso_regression", las)])

    predictions = cross_val_predict(estimator, X, y, cv = kf)

    score = r2_score(y, predictions)

    scores.append(score)
```

In [49]:

```
plt.figure(figsize=(10,6))
plt.semilogx(alphas, scores, '-o')
plt.xlabel('$\\alpha$')
plt.ylabel('$R^2$');
```



In [ ]:

```

```

In [50]:

```
alpha = max(list(zip(alphas,scores)))[1] #We get the better hyperparameter from the  R2

lasso_score_list = list(zip(alphas,scores))
for i in range (len(score_list)):
    if lasso_score_list[i][1] ==max(scores):
        print(lasso_score_list[i][1])
        alpha = lasso_score_list[i][0]
```

```
0.8590101013738956
```

In [58]:

```
alpha
```

Out[58]:

```
0.1
```

In [51]:

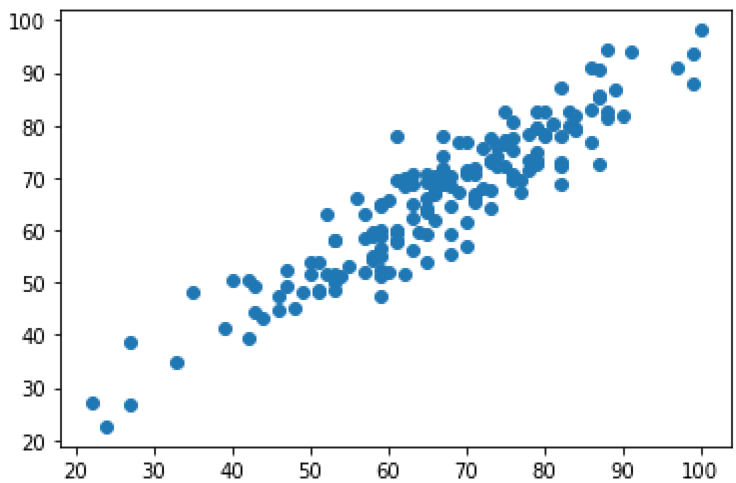
```
#Initialize objects
s = StandardScaler()
lr = LinearRegression()
pf = PolynomialFeatures(degree = 3)
las = Lasso(alpha=alpha, max_iter=100000)
best_estimator = Pipeline([
        ("poly features", pf),
        ("scaler", s),
        ("lasso_regression", las)])
```

In [52]:

```
best_estimator.fit(X_train, y_train)
best_estimator.predict(X_test)
plt.scatter(y_test, best_estimator.predict(X_test))
```

Out[52]:

```
<matplotlib.collections.PathCollection at 0x244a1c1f8b0>
```



In [53]:

```
best_predictions = cross_val_predict(best_estimator, X, y, cv=kf)
```

In [54]:

```
r2_score(y, best_predictions)
```

Out[54]:

```
0.8590101013738956
```

In [63]:

```
pd.DataFrame(zip(X.columns, las.coef_)).sort_values(by=1)
```

Out[63]:

	0	1
10	parental_level_of_education_some college	-0.137959
0	reading_score	0.000000
4	race_ethnicity_group C	-0.000000
5	race_ethnicity_group D	-0.000000
6	race_ethnicity_group E	0.000000
7	parental_level_of_education_bachelor's degree	0.000000
8	parental_level_of_education_high school	-0.000000
11	parental_level_of_education_some high school	-0.000000
12	lunch_standard	-0.000000
9	parental_level_of_education_master's degree	0.089780
13	test_preparation_course_none	0.129093
1	writing_score	1.830519
3	race_ethnicity_group B	4.769109
2	gender_male	10.386407