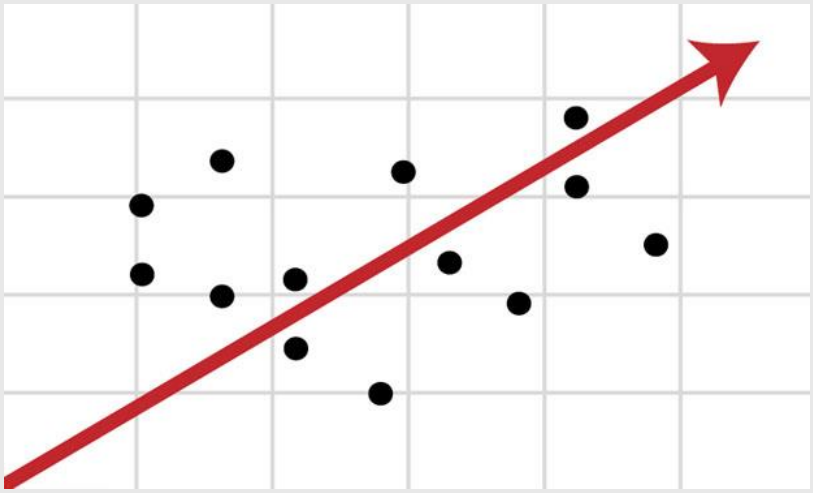
PROJECT 3: REGRESSION ANALYSIS



Jesus Barrera Mejia

ITCS 3612 002

Professor Aileen

3/22/2024

The Problem & The Data

This project will focus on answering the question “What impacts university graduation rates?” The dataset was retrieved from Kaggle has 62 features and was a part of a package with 3 more datasets. For this project, we will focus solely on the dataset that has information about the institutions and the graduation rates of those institutions. Also, in the dataset, we’re given two graduation rates, grad_100_value & grad_150_value. Grad 100 is the graduation rate with an upper bound of 4 years as in the people tracked complete their degree within 4 years. Grad150 are the people who completed their degree in 5 years and includes those from grad 100 too. Additionally, these values only track a small population and there is a possibility of sample bias since the people being tracked are traditional students: excludes transfer students, non-traditional students, students who changed their field of study, and more. Although there are 62 features, I only use 15. Here are the features.

`state`: I feel like using state we can gain insight into diffrent trends based on region
`level` - Level of institution (4-year, 2-year)
`control`: Control is the classification if the school is Public, private non-profit, or private for-profit
`hbcu`: Historically Black College
`flagship`: Basically if the school is a "Flagship" insitituation (ex. UNC, University of Michigan)
`student_count` - Total number of undergraduates in 2010
`exp_award_value`: The amount of educational sepnding per award (degree/certificate)
`retain_value`: Share of freshmen retained for a second year
`pell_value`: percentage of undergraduates receiving a Pell Grant
`ft_fac_value`: Percentage of employees devoted to instruction, research or public service who are full-time and do not work for an associated medical school
`counted_pct`: share entering undergraduate class who were first-time, full-time, degree-seeking students, meaning that they generally would be part of a tracked cohort of potential graduates. The entering class of 2007 is displayed for 4-year institutions; 2010 for 2-year institutions.
`aid_value` - The average amount of student aid going to undergraduate recipients

Our dependent variable will be.

`grad_150_value`: percentage of first-time, full-time, degree-seeking undergraduates who complete a degree or certificate program within 150 percent of expected time (bachelor's-seeking group at 4-year institutions)

What Is Regression?

Regression is a statistical method that we use to make quantitative predictions. For example, “How much does price increase per increase in quantity?” The linear regression model makes these predictions using a **line of best fit**. The math to find the OLS is dividing the sum of the variations in X multiplied by the sum in variation in y by the **total sum of squares** of x. Here is an illustration.

$$m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$
$$b = \bar{y} - m * \bar{x}$$

$x = independent\ variables$
$\bar{x} = average\ of independent\ variables$
$y = dependent\ variables$
$\bar{y} = average\ of dependent\ variables$

We measure how well of a predictor the **line of best fit** is by using **R-squared** which is calculated as the sum of errors minus the sum of errors for the line of best fit divided by the sum of errors of the mean. We are essentially measuring how much of the variation is captured by the line compared to just using the mean of the data. Here’s an illustration of the formula.

$$R^2 = \frac{SS(mean) - SS(fit)}{SS(mean)}$$

Data Understanding

To understand my data my first step was to read information from the source. Fortunately, we were given a legend for the features, so I began by learning about what each feature represented. Based on that information I compiled 15 features to make my project interpretable. Secondly, I began to investigate my features, how the data was formatted, and some basic statistics for them. For my dependent variable, I found that the mean grad 150 value was 42%. On average schools had 4476 students, spent $65074.47 per award (including certificates), retained 66% of freshmen, had 48% of students receiving Pell grants, 45% of staff as full time, and provided students $7960 in aid. However, this is only round one of statistics and does not include all features, so we must do some **preprocessing** to gain more insight.

	student_count	exp_award_value	retain_value	pell_value	ft_fac_value	aid_value	grad_100_value	grad_150_value
count	3798.000000	3.798000e+03	3535.000000	3797.000000	3785.000000	3797.000000	3467.000000	3467.000000
mean	4476.135334	6.507447e+04	66.231853	47.572057	45.107477	7960.445878	28.364465	42.407586
std	7376.868923	1.074379e+05	17.033907	20.065216	24.726902	6419.658196	23.312730	23.460824
min	23.000000	0.000000e+00	0.000000	0.000000	0.000000	294.000000	0.000000	0.000000
25%	581.250000	3.231125e+04	56.100000	32.400000	25.700000	4018.000000	9.000000	22.700000
50%	1794.500000	5.057850e+04	66.900000	44.700000	41.500000	5207.000000	22.500000	41.100000
75%	5172.000000	7.693025e+04	78.100000	62.500000	63.000000	9343.000000	43.650000	60.250000
max	170144.000000	5.282095e+06	100.000000	100.000000	100.000000	41580.000000	100.000000	100.000000

Preprocessing + More Understanding

Data Types & Buckets

Using the dtypes function we discover which features are categorical and will require further formatting. We find state, level, control, hbcu, & flagship will need to be one-hot encoded. Secondly, we use the .value_counts() function to find out which buckets

each feature has (cell 10, Project 3 Notebook). Using value_counts(), we find out that our HBCU & flagship features use nulls to describe entries that do not classify as those features, we will fix this using the replace() function; Secondly, we will have to one-hot encode **level & control**. For **level**, we can change the feature to be **is_four_year** and one hot encode it using the **replace()** function; Lastly, for **state,** I wanted to gain more insight, so I decided to replace **state** with **region** – West, Northeast, south, and categorize that data that way.

```
chronname          object
state              object
level              object
control            object
hbcu               object
flagship           object
student_count       int64
exp_award_value     int64
retain_value      float64
pell_value        float64
ft_fac_value      float64
aid_value         float64
counted_pct        object
grad_100_value    float64
grad_150_value    float64
```

```
for col in dataSet2:
    print(f'{col}:\n{dataSet2[col].value_counts()}')
```

Null Values

```
dataSet2.isnull().sum()
[11]  ✓ 0.0s
...
chronname            0
state                0
level                0
control              0
hbcu              3704
flagship          3748
student_count        0
exp_award_value      0
retain_value       263
pell_value           1
ft_fac_value        13
aid_value            1
counted_pct        426
grad_100_value     331
grad_150_value     331
dtype: int64
```

The dataset had 9 features with null values, however, it's important to note that 2 used null values to classify the data – **hbcu & flagship**. We begin by replacing the null values in **hbcu & flagship** with 0 and replacing the X they contain with 1 starting in cell 13 of the notebook. I also decided to drop **counted_pct** because it was oddly formatted. For the rest of the nulls, we use the dropna() function.

One-Hot Encoding

```
def find_region(state):
    for region, states in regions.items():
        if state in states:
            return region
    return 0
```

While fixing null values, we also one-hot encoded **hbcu & flagship**, so we have **state**, **level**, & **control** left to format. For **state**, I created a dictionary to classify each state by region and created a function to classify each entry based on **region** in cell 17. In cell 19, I ran the function using a **for loop** & in line 20 I dropped **Midwest** to prevent **multicollinearity** in our model. Lastly, for **level,** I used the **replace()** function and one-hot encode the column to **is_four_year** where 1 means yes and 0 means no (Cell 22). For **control**, I used the **get_dummies()** function because there are more than two buckets and drop **private for-profit schools** leaving us with **private not-for-profit** & **public** (Cell 24).

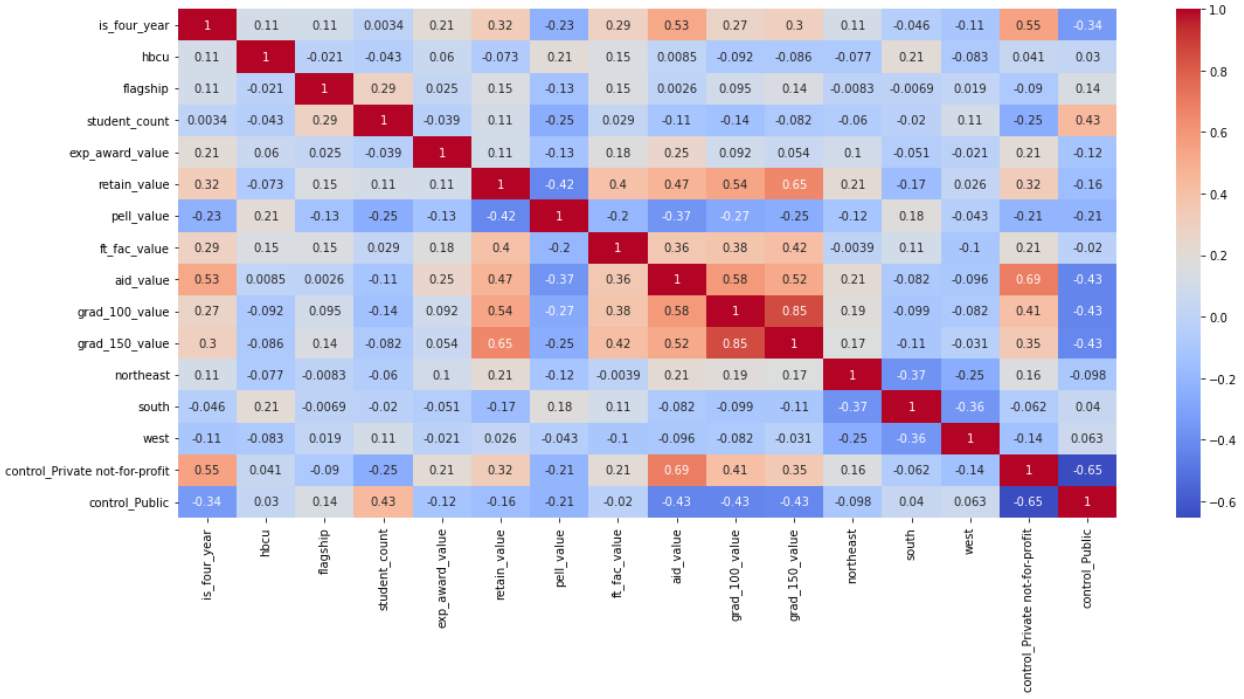
Formatting

```
dataSet2["student_count"] = dataSet2["student_count"]/100
dataSet2["exp_award_value"] = dataSet2["exp_award_value"]/100
dataSet2["aid_value"] = dataSet2["aid_value"]/100
```

[24] ✓ 0.0s

Our final preprocessing step involves formatting numbers for the sake of interpretability. In this step, we divide the **student_count**, **exp_award_value**, and **aid_value** by 100. This allows us to express these values on a per 100 student and per $100 basis, which will be beneficial when reviewing the **statsmodels** summary.

Understanding - Part 2



With the pre-processed dataset, we gain new insights. We observe a positive correlation between our dependent variable with both retain_value and ft_fav_value. Additionally, there's a strong correlation between is_four_year and Private not-for-profit,

indicating that private not-for-profit schools are typically four-year institutions. Importantly, our dataset does not exhibit multicollinearity, so we can commence with our modeling.

Experiment 1 – Standardized Data vs. Non-Standardized Data

For Experiment 1, I wanted to test the impact that standardization has on a model's predictive value. Cells 31 to 33 include setting up the variables, splitting the data, and creating standardized variables.

Non-standardized: Stats Model

OLS Regression Results

Dep. Variable:	grad_150_value	R-squared:	0.634
Model:	OLS	Adj. R-squared:	0.632
Method:	Least Squares	F-statistic:	289.6
Date:	Fri, 22 Mar 2024	Prob (F-statistic):	0.00
Time:	13:03:40	Log-Likelihood:	-9584.7
No. Observations:	2359	AIC:	1.920e+04
Df Residuals:	2344	BIC:	1.929e+04
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	8.2900	2.317	3.578	0.000	3.747	12.833
is_four_year	-2.0432	0.753	-2.715	0.007	-3.519	-0.567
hbcu	-2.6065	2.000	-1.304	0.193	-6.528	1.315
flagship	15.2896	2.499	6.117	0.000	10.388	20.191
student_count	-0.0084	0.004	-1.964	0.050	-0.017	-1.32e-05
exp_award_value	-0.0018	0.000	-7.577	0.000	-0.002	-0.001
retain_value	0.6554	0.023	27.999	0.000	0.609	0.701
pell_value	-0.1286	0.021	-6.243	0.000	-0.169	-0.088
ft_fac_value	0.2107	0.014	14.984	0.000	0.183	0.238
aid_value	0.0686	0.007	9.871	0.000	0.055	0.082
northeast	0.1564	0.882	0.177	0.859	-1.574	1.887
south	-1.1098	0.785	-1.414	0.158	-2.649	0.430
west	-1.2490	0.897	-1.392	0.164	-3.009	0.511
control_Private not-for-profit	-13.6999	1.095	-12.511	0.000	-15.847	-11.553
control_Public	-23.3920	0.965	-24.240	0.000	-25.284	-21.500

Omnibus:	172.760	Durbin-Watson:	1.980
Prob(Omnibus):	0.000	Jarque-Bera (JB):	884.729
Skew:	-0.065	Prob(JB):	7.65e-193
Kurtosis:	5.997	Cond. No.	1.29e+04

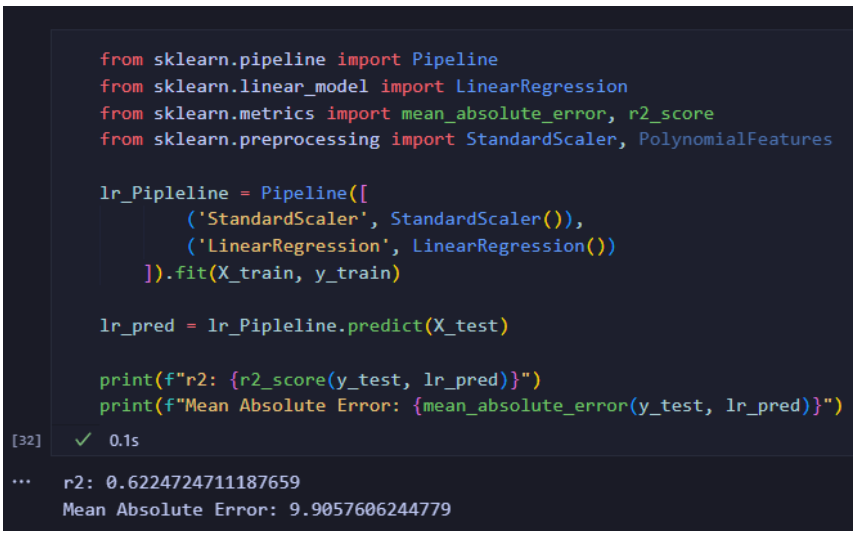
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.29e+04. This might indicate that there are strong multicollinearity or other numerical problems.

We begin with the non-standardized **linear regression** using the **statsmodels** package. The benefit of using statsmodels without standardization is we can interpret our variables: For example, *flagship schools have a grad_150 value of 15.29 percentage points*

higher compared to non-flagship schools. Additionally, the summary shows us if our features are **statistically significant.**

The model shows us that the following features are significant: Is_four_year, Flagship, Student Count, Exp_award_value, Retain_value, Pell_value, Ft_fac_value, aid_value, control_Private not-for-profit, control_Publicj.

Standardized Model: Scikit Learn

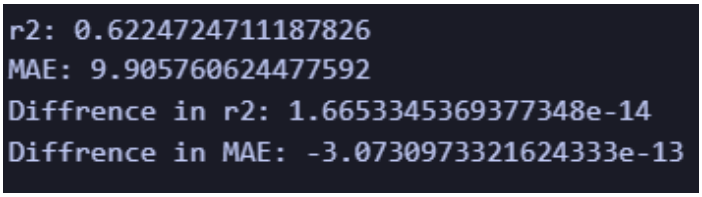


```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler, PolynomialFeatures

lr_Pipleline = Pipeline([
        ('StandardScaler', StandardScaler()),
        ('LinearRegression', LinearRegression())
    ]).fit(X_train, y_train)

lr_pred = lr_Pipleline.predict(X_test)

print(f"r2: {r2_score(y_test, lr_pred)}")
print(f"Mean Absolute Error: {mean_absolute_error(y_test, lr_pred)}")
```

```
[32]  ✓ 0.1s

...  r2: 0.6224724711187659
     Mean Absolute Error: 9.9057606244779
```

Scikit's Linear Regression is simpler than statsmodels. With Scikit we build the model and then make predictions, however, it's less interpretable since we can't print the results.

Evaluation



```
r2: 0.6224724711187826
MAE: 9.905760624477592
Diffrence in r2: 1.6653345369377348e-14
Diffrence in MAE: -3.0730973321624333e-13
```

In cell 34, to evaluate both models, I use the statsmodels linear regression to make predictions. I provide the **R-squared**, but MAE (**Mean Absolute Error)** is a better measure for predictive value. We discover that the **Standardized** model has a lower **MAE** than the **non-**

standardized model, but the difference is 13 decimal places. It's a small difference, but an interesting one considering that we standardize data to reduce error.

Experiment 2 – Adding a Polynomial Feature

```
Number of polynomials:1
r2: 0.6224724711187659
Mean Absolute Error: 9.905760624477896

Number of polynomials:2
r2: 0.6757713954137903
Mean Absolute Error: 8.721436318861167

Number of polynomials:3
r2: -318294103540707.56
Mean Absolute Error: 19446234.829054564

Number of polynomials:4
r2: -153835036440058.7
Mean Absolute Error: 29218293.650210895

Number of polynomials:5
r2: -21522637.84059453
Mean Absolute Error: 5030.56571756194
```

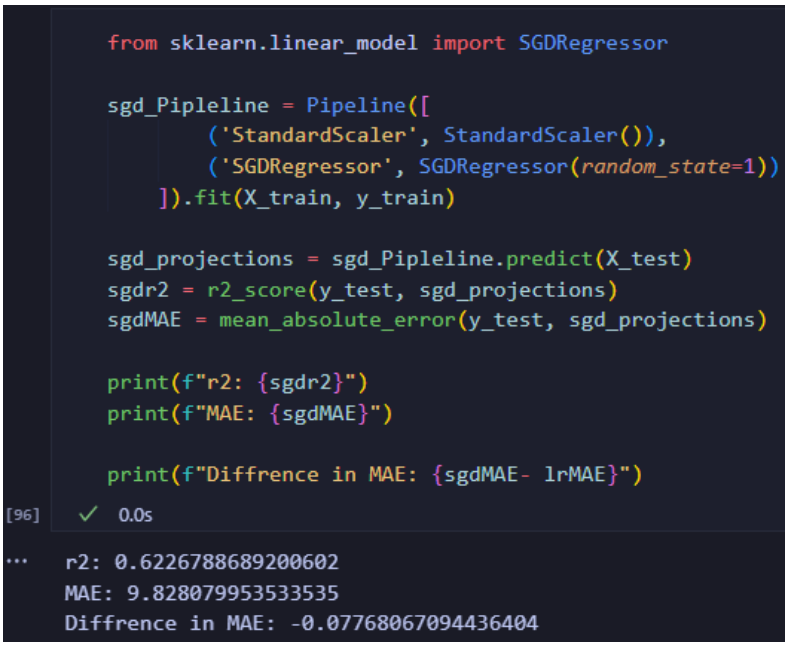
In the next experiment, we add a polynomial to our experiment and compare it with the standardized model from before. Adding a polynomial allows our model to have curves in the slope and capture more information. For this process, I created a loop in cell 35 to find the optimal number of polynomials which was 2.

Experiment 2 - Evaluation

```
    8.721436318861167 - lrMAE
[91] ✓ 0.0s
...  -1.184324305616732
```

When we subtract the **MAE** of the model with polynomials from the **MAE** of the regular linear regression, we find that the polynomials can capture more information and give us an **MAE** that's lower by 1.18.

Experiment 3 – Stochastic Gradient Decent



```
from sklearn.linear_model import SGDRegressor

sgd_Pipleline = Pipeline([
        ('StandardScaler', StandardScaler()),
        ('SGDRegressor', SGDRegressor(random_state=1))
    ]).fit(X_train, y_train)

sgd_projections = sgd_Pipleline.predict(X_test)
sgdr2 = r2_score(y_test, sgd_projections)
sgdMAE = mean_absolute_error(y_test, sgd_projections)

print(f"r2: {sgdr2}")
print(f"MAE: {sgdMAE}")

print(f"Diffrence in MAE: {sgdMAE- lrMAE}")
```

```
[96]  ✓ 0.0s
...   r2: 0.6226788689200602
      MAE: 9.828079953533535
      Diffrence in MAE: -0.07768067094436404
```

The **Stochastic Gradient Decent regressor** is a regressor that minimizes the **loss function** to improve its predictions. The loss function is a curved line and by incrementally traversing that slope we find the minimum. We're essentially finding the coefficient for our independent variables that minimize the loss functions instead of calculating it ourselves using the OLS function.

Experiment 3 – Evaluation

We find that by using the **Stochastic Gradient Decent regressor** we improve our error by 0.08, a small difference compared to adding polynomials. This shows us that adding polynomials helps us capture more information compared to optimizing the loss function.

Impact

This project may have a positive impact by helping colleges understand what they can do to ensure that their students are graduating. Additionally, it can help students choose schools that will help them graduate and provide them insight into what features they should pay attention to for graduation rates. Conversely, the project may also be misleading considering that the graduation rate provided by the schools only includes a small portion of the students who are graduating and excludes non-traditional students.

Conclusion

During this project, I sharpened my pre-processing skills, learned a valuable lesson about the importance of having a good dataset, and learned about how using different models and techniques can impact the performance of machine learning models. For preprocessing, I took time to learn about my features. I found that most statistical relationships and correlation can be explained when you know what your features are: For example, grad_150 had a high correlation with grad_100, but the explanation provided for the features shows us that this is because grad_150 includes grad_100. For the models, I learned that it's important to try different techniques with your models. In my model, the use of polynomial features enhanced the information captured by the models. However, not all modifications yield significant impacts, but it's crucial to experiment. For instance, the application of the stochastic gradient descent technique slightly improved the model's error, similar to the minor changes observed when switching between standardized and non-standardized data.

Refrences

1. The following was for an error that I had when importing the data:

The error was: UnicodeDecodeError: 'utf-8' codec can't decode byte 0xff in position 0: invalid start byte

https://stackoverflow.com/questions/42339876/error-unicodedecodeerror-utf-8-codec-cant-decode-byte-0xff-in-position-0-in

2. I used stats models to figure out how to create my regression models:

https://www.statsmodels.org/stable/generated/statsmodels.regression.linear_model.OLS.html

3. Used the following to help me understand the pipeline function from scikit and what is going on:

https://www.youtube.com/watch?v=jzKSAeJpC6s