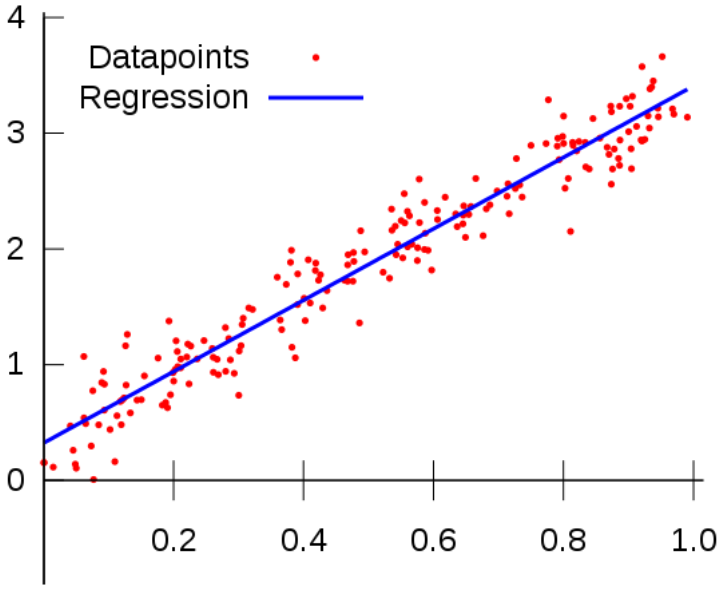
Exploratory Data Analysis and Linear Regression



Today, we will learn how to use EDA (Exploratory Data Analysis) along with linear regression to predict graduation rate in Hartford, CT. Let's get started!

Environment Specifications

We will be using TensorFlow for this tutorial. If you have not downloaded TensorFlow on your system please follow these steps:

1) Download Anaconda: https://www.anaconda.com/distribution/ (https://www.anaconda.com/distribution/)

2) Run `conda install tensorflow` from the command line.

Great! Now that you have TensorFlow installed we can get started.

INTRODUCTION

What is TensorFlow?

So, what is TensorFlow and how are we going to use it? As it is described by Google, "TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms." (https://www.tensorflow.org/about/bib (https://www.tensorflow.org/about/bib)). Using TensorFlow's predefined functions, we will create a structure to implement different machine learning methods - specifically in this tutorial, we will use machine learning to create a linear regression model.

- For more information on TensorFlow, click here: Get started with TensorFlow (https://www.tensorflow.org/tutorials/)

What is EDA?

"EDA or Exploratory Data Analysis is an approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to

- maximize insight into a data set;
- uncover underlying structure;
- extract important variables;
- detect outliers and anomalies;
- test underlying assumptions;
- develop parsimonious models; and
- determine optimal factor settings." (https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm (https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm))

Further, we can explore data to find important correlations between variables and use those correlations to build relevant models.

What is Linear Regression?

In simple terms, Linear Regression is a way to predict a certain "Y" (outcome) from a given "X" (predictor variable). When there is a linear relationship between two variables, we can use machine learning to create this kind of model.

"Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things: (1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable? (2) Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable?" (https://www.statisticssolutions.com/what-is-linear-regression/ (https://www.statisticssolutions.com/what-is-linear-regression/))

How are we going to use both EDA and TensorFlow to do Linear Regression?

1) Explore our dataset using EDA to find correlations between variables

2) Once we find our variables, determine if they have a linear relationship

3) If they have a linear relationship, split the data up into train and test sets

4) Using TensorFlow, create a mathematical model that represents our data

5) Use linear regression on our model to make predictions about our linear variables

EDA

An Overview

As stated by Chong Ho Yu in his paper *Exploratory Data Analysis*, "Exploratory data analysis (EDA) is a strategy of data analysis that emphasizes maintaining an open mind to alternative possibilities. EDA is a philosophy or an attitude about how data analysis should be carried out, rather than being a fixed set of techniques." (http://www.creative-wisdom.com/teaching/551/Reading_materials/Yu_EDA_Oxford.pdf (http://www.creative-wisdom.com/teaching/551/Reading_materials/Yu_EDA_Oxford.pdf))

EDA is an idea that we carry out rather than a specific toolset we use to find important data. While EDA does not define a certain set of techniques for us to use to explore data, there are some useful tools we can manipulate our data with to help in the exploration process!

Programming Tools

EDA is a hard thing to do by just looking at data with the eye! Thankfully, there are many programming languages that are extremely useful for manipulating data. One of them is Python!

Specifically, we will be using a Python Data Science library called *Pandas*.

"Python has long been great for data munging and preparation, but less so for data analysis and modeling. pandas helps fill this gap, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R." (https://pandas.pydata.org/ (https://pandas.pydata.org/))

Pandas is a great tool for EDA and we will learn more about it later in this tutorial.

Why EDA is Useful and Things we Need to Look out for

"Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations." (https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15 (https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15))

Because EDA is so important, we need to keep a couple of things in mind:

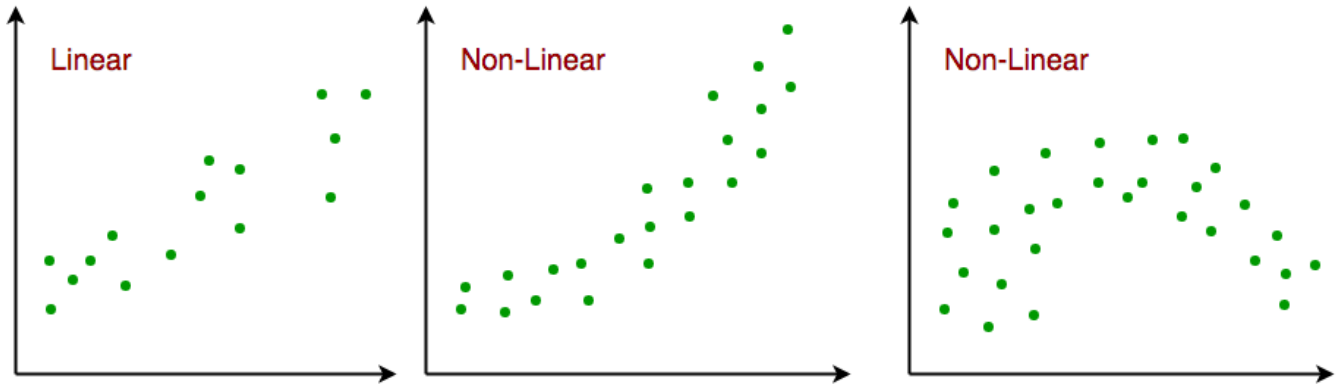
Why is EDA Useful?

- Shows us correlations between variables
- Helps us pick out certain parts of the data that are useful
- Alerts us of erroneous parts of the data, and helps us remove it

What should we look out for?

- Removing important aspects of the data
- Making sure there is a correlation between varibales before assuming causation
- Bias in data

LINEAR REGRESSION



An Overview

Ever wish you could predict the future? Well with linear regression, now you can! Linear regression helps us predict outcomes from certain variables.

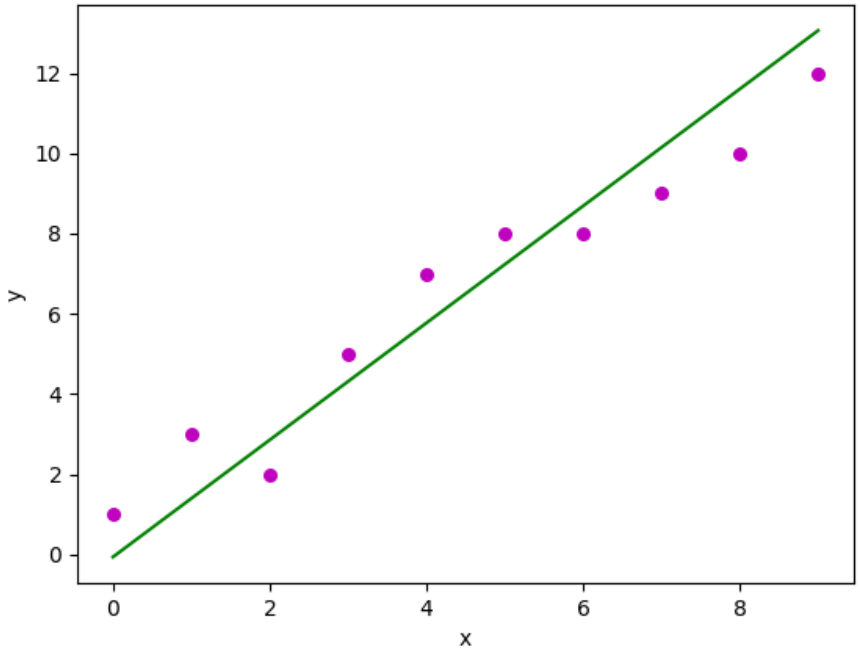
"Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.

Before attempting to fit a linear model to observed data, a modeler should first determine whether or not there is a relationship between the variables of interest. This does not necessarily imply that one variable causes the other (for example, higher SAT scores do not cause higher college grades), but that there is some significant association between the two variables. A scatterplot can be a helpful tool in determining the strength of the relationship between two variables. If there appears to be no association between the proposed explanatory and dependent variables (i.e., the scatterplot does not indicate any increasing or decreasing trends), then fitting a linear regression model to the data probably will not provide a useful model. A valuable numerical measure of association between two variables is the correlation coefficient, which is a value between -1 and 1 indicating the strength of the association of the observed data for the two variables.

A linear regression line has an equation of the form `Y = a + bX` , where `X` is the explanatory variable and `Y` is the dependent variable. The slope of the line is `b` , and a is the intercept (the value of `y` when `x = 0`)." (http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm (http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm))

How Linear Regression is Used for Prediction

A regression line is used to precict Y from X. When there is a linear relationship between two variables we can draw a regression line through the data, like the photo below.



The regression line is green and the data points are purple. If our `X` value was `2`, our regression line would predict a `Y` value of `2.7`.

How do we draw an accurate regression line? Well, there's a formula!

"In the age of computers, the regression line is typically computed with statistical software. However, the calculations are relatively easy, and are given here for anyone who is interested. M_X is the mean of X, M_Y is the mean of Y, s_X is the standard deviation of X, s_Y is the standard deviation of Y, and r is the correlation between X and Y.

The formula for standard deviation `s` is:

$$\sigma = \sqrt{\frac{\Sigma\ (X - \overline{X})^2}{n - 1}}$$

The formula for correlation `r` is:

$$r = \frac{\sum(x-\overline{x})(y-\overline{y})}{\sqrt{\sum(x-\overline{x})^2 \sum(y-\overline{y})^2}}$$

The slope (b) can be calculated as follows:

$b = r\ s_Y/s_X$

and the intercept (A) can be calculated as

$A = M_Y - bM_X$." (http://onlinestatbook.com/2/regression/intro.html (http://onlinestatbook.com/2/regression/intro.html))

Great! Now we know how to plot a regression line.

DATA

Connecticut School Accountability Data

We will be using the State Department of Education Next Generation School Accountability data. Here's some information about it!

"Connecticut's Next Generation Accountability System is a broad set of 12 indicators that help tell the story of how well a school is preparing its students for success in college, careers and life. The new system moves beyond test scores and graduation rates and instead provides a more holistic, multifactor perspective of district and school performance and incorporates student growth over time. The 12 Indicators include:

1. Academic achievement status measured by state assessments
2. Academic growth
3. Assessment participation rate
4. Chronic absenteeism
5. Preparation for postsecondary and career readiness – coursework
6. Preparation for postsecondary and career readiness – exams
7. Graduation – on track in ninth grade
8. Graduation – four-year adjusted cohort graduation rate – all students
9. Graduation – six-year adjusted cohort graduation rate – high needs
10. Postsecondary entrance rate – all students (college enrollment)
11. Physical fitness
12. Arts access" Source: https://data.ct.gov/Education/State-Department-of-Education-Next-Generation-Scho/mdsf-2nsf/data (https://data.ct.gov/Education/State-Department-of-Education-Next-Generation-Scho/mdsf-2nsf/data)

How are we Going to Use the Data?

We will be using this data to predict graduation rate from all of these metrics!

- First, we will use EDA to find which variables have a linear relationship with graduation rate.
- Next, we will train a model with our data.
- Finally, we will test our model to see if our predictions were accurate.

TUTORIAL

EDA

Let's do some EDA to understand our dataset better.

First, we will load our data with Python's Pandas library. Let's import Pandas, Numpy, and TensorFlow, as we will need them throughout the tutorial.

In [1]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
```

Great! Now that we have everything imported, let's look at our data! We will use pandas `pd.read_csv()` function to read in our data into a table called a DataFrame. For more information about pandas, check out the documentation here: https://pandas.pydata.org/pandas-docs/stable/ (https://pandas.pydata.org/pandas-docs/stable/)

We will store our Connecticut School Data in a DataFrame called `CT_school_data` .

Note: Calling `head()` on a DataFrame will show us the DataFrame's first 5 rows.

In [2]:
```
CT_school_data = pd.read_csv('Data.csv')

CT_school_data.head()
```

Out[2]:

	District Name	District Code	School Name	School Code	School OrgType	School Low Grade	School High Grade	School Title I Type	Category	P
0	Andover School District	10011	District_0000000	0	District	District	NaN	NaN	DistrictTot	6
1	Andover School District	10011	Andover Elementary School_0010111	10111	Public Schools	PK	6.0	Targeted Assistance	SchoolTot	6
2	Ansonia School District	20011	District_0000000	0	District	District	NaN	NaN	DistrictTot	8
3	Ansonia School District	20011	Mead School_0020311	20311	Public Schools	PK	6.0	Schoolwide	SchoolTot	4
4	Ansonia School District	20011	Prendergast School_0020811	20811	Public Schools	K	6.0	Schoolwide	SchoolTot	5

5 rows × 88 columns

As you can see, our dataset has 88 columns. Let's checkout the column names so we can start deciding which variables will be important to us.

In [3]:
```
CT_school_data.columns
```

Out[3]:
```
Index(['District Name', 'District Code', 'School Name', 'School Code',
       'School OrgType', 'School Low Grade', 'School High Grade',
       'School Title I Type', 'Category', 'Total Points',
       'Total Possible Points', 'Accountability Index', 'Achievement Ga
p Flag',
       'Ind1 ELA_All_Rate', 'Ind1 ELA_All_Points',
       'Ind1 ELA_All_PossiblePoints', 'Ind1 ELA_HN_Rate', 'Ind1 Math_NH
N_Rate',
       'Ind1 ELA_NHN_Rate', 'Ind1 ELA_HN_Points', 'Ind1ELA_HN_PossibleP
oints',
       'Ind1Math_All_Rate', 'Ind1Math_All_Points',
       'Ind1Math_All_PossiblePoints', 'Ind1Math_HN_Rate', 'Ind1Math_HN_
Points',
       'Ind1Math_HN_PossiblePoints', 'Ind1Sci_All_Rate', 'Ind1Sci_All_P
oints',
       'Ind1Sci_All_PossiblePoints', 'Ind1Sci_HN_Rate', 'Ind1Sci_NHN_Ra
te',
       'Ind1Sci_HN_Points', 'Ind1Sci_HN_PossiblePoints', 'Ind1ELAGap',
       'Ind1MathGap', 'Ind1SciGap', 'Ind3PartRateFlag', 'Ind3ELA_All_Ra
te',
       'Ind3ELA_HN_Rate', 'Ind3Math_All_Rate', 'Ind3Math_HN_Rate',
       'Ind3Sci_All_Rate', 'Ind3Sci_HN_Rate', 'Ind4Rate', 'Ind4Points',
       'Ind4PossiblePoints', 'Ind4HNRate', 'Ind4HNPoints',
       'Ind4HNPossiblePoints', 'Ind5Rate', 'Ind5Points', 'Ind5PossibleP
oints',
       'Ind6Rate', 'Ind6Points', 'Ind6PossiblePoints', 'Ind7Rate',
       'Ind7points', 'Ind7PointsPossible', 'Ind8Rate', 'Ind8Points',
       'Ind8PointsPossible', 'Ind9Rate', 'Ind9Points', 'Ind9PointsPossi
ble',
       'GradGapFlag', 'Ind10Rate', 'Ind10Points', 'Ind10PossiblePoint
s',
       'Ind11FitnessRate', 'Ind11ParticipationRate', 'Ind11Points',
       'Ind11PossiblePoints', 'Ind12Rate', 'Ind12Points',
       'Ind12PossiblePoints', 'Grad6YrRateHN', 'Grad6YrRateNHN',
       'Grad6YrHNNHNDiff', 'Threshold_Mean_ELA', 'Threshold_Mean_Math',
       'Threshold_Mean_Sci', 'Threshold_Mean_Grad', 'Classification',
       'State Category', '# of Distinctions',
       'School of Distinction - Highest Performing Overall',
       'School of Distintion - Highest Performing High Needs'],
      dtype='object')
```

We need to predict graduation rate from one of these variables. This means that we need the graduation rate from this data. It looks like the column name `Grad6YrRateHN` is related to some sort of graduation rate. Let's check it out and see how it looks.

In [4]:
```
CT_school_data[['Grad6YrRateHN']].head()
```

Out[4]:

	Grad6YrRateHN
0	NaN
1	NaN
2	79.7
3	NaN
4	NaN

Hmmm, something looks odd. There is only one graduation rate! The rest of the first five are `NaN` values, which means we have some missing data! Let's see how many graduation rates have an integer as their value (not null) and subtract it from the number of graduation rates that have `NaN` as their value.

In [5]:
```
len(CT_school_data) - len(CT_school_data[CT_school_data['Grad6YrRateHN']
.isnull()])
```

Out[5]: 300

There are 300 integer values which means we have 300 graduation rates to work with. This will be more than enough to train our data.

Let's filter our original data to only contain graduation rates without `NaN` values.

In [6]:
```
filtered_data = CT_school_data[CT_school_data['Grad6YrRateHN'].isnull()
== False]

filtered_data.head()
```

Out[6]:

	District Name	District Code	School Name	School Code	School OrgType	School Low Grade	School High Grade	School Title I Type	Category	P
2	Ansonia School District	20011	District_0000000	0	District	District	NaN	NaN	DistrictTot	8
6	Ansonia School District	20011	Ansonia High School_0026111	26111	Public Schools	9	12.0	NaN	SchoolTot	8
9	Avon School District	40011	District_0000000	0	District	District	NaN	NaN	DistrictTot	1
14	Avon School District	40011	Avon High School_0046111	46111	Public Schools	9	12.0	Targeted Assistance	SchoolTot	1(
18	Berlin School District	70011	District_0000000	0	District	District	NaN	NaN	DistrictTot	1(

5 rows × 88 columns

Great! Now our data only contains rows with graduation rates.

Looking further, if we want to localize our data to certain areas in CT, we can use the `District Code` column to choose certain schools in a specific area. If we wanted to only look at schools in Hartford, CT we could choose only Hartford, CT District Codes. Listed below are District Codes for schools in Hartford, CT. Let's filter our data using these codes so we are left with schools that are in Hartford, CT.

In [7]:
```
Hartford = np.array([540011, 640011, 1280011, 40011, 520011, 1550011, 13
20011, 230011, 560011, 1390011, 1190011, 1590011, 400011, 940011, 70011,
170011, 1100011, 1310011, 1640011])

Hartford_data = filtered_data[filtered_data['District Code'].isin(Hartfo
rd)]

Hartford_data.head()
```

Out[7]:

	District Name	District Code	School Name	School Code	School OrgType	School Low Grade	School High Grade	School Title I Type	Category	P
9	Avon School District	40011	District_0000000	0	District	District	NaN	NaN	DistrictTot	11
14	Avon School District	40011	Avon High School_0046111	46111	Public Schools	9	12.0	Targeted Assistance	SchoolTot	10
18	Berlin School District	70011	District_0000000	0	District	District	NaN	NaN	DistrictTot	10
23	Berlin School District	70011	Berlin High School_0076111	76111	Public Schools	9	12.0	Targeted Assistance	SchoolTot	9
91	Bristol School District	170011	District_0000000	0	District	District	NaN	NaN	DistrictTot	9

5 rows × 88 columns

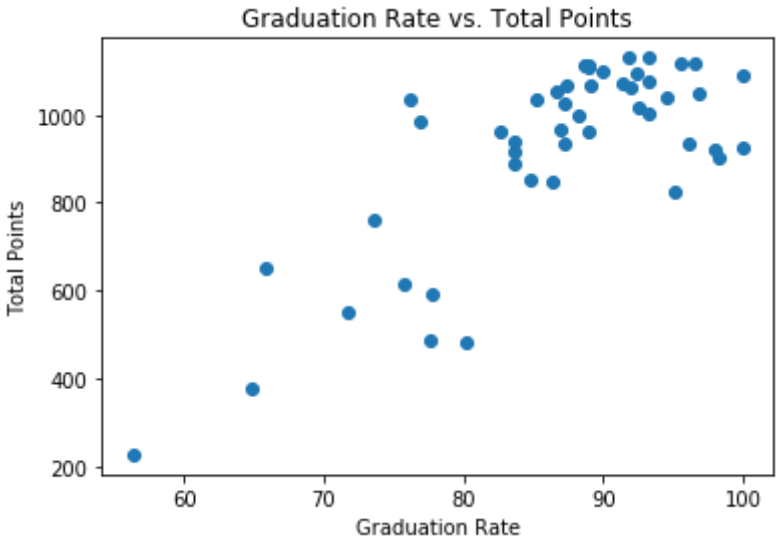
Great! Now that we have data for just Hartford, CT let's choose variables that are linear with graduation rate. Let's look at the columns again so we can infer which ones we might think will be linear with graduation rate.

In [8]:
```
Hartford_data.columns
```

Out[8]:
```
Index(['District Name', 'District Code', 'School Name', 'School Code',
       'School OrgType', 'School Low Grade', 'School High Grade',
       'School Title I Type', 'Category', 'Total Points',
       'Total Possible Points', 'Accountability Index', 'Achievement Ga
p Flag',
       'Ind1 ELA_All_Rate', 'Ind1 ELA_All_Points',
       'Ind1 ELA_All_PossiblePoints', 'Ind1 ELA_HN_Rate', 'Ind1 Math_NH
N_Rate',
       'Ind1 ELA_NHN_Rate', 'Ind1 ELA_HN_Points', 'Ind1ELA_HN_PossibleP
oints',
       'Ind1Math_All_Rate', 'Ind1Math_All_Points',
       'Ind1Math_All_PossiblePoints', 'Ind1Math_HN_Rate', 'Ind1Math_HN_
Points',
       'Ind1Math_HN_PossiblePoints', 'Ind1Sci_All_Rate', 'Ind1Sci_All_P
oints',
       'Ind1Sci_All_PossiblePoints', 'Ind1Sci_HN_Rate', 'Ind1Sci_NHN_Ra
te',
       'Ind1Sci_HN_Points', 'Ind1Sci_HN_PossiblePoints', 'Ind1ELAGap',
       'Ind1MathGap', 'Ind1SciGap', 'Ind3PartRateFlag', 'Ind3ELA_All_Ra
te',
       'Ind3ELA_HN_Rate', 'Ind3Math_All_Rate', 'Ind3Math_HN_Rate',
       'Ind3Sci_All_Rate', 'Ind3Sci_HN_Rate', 'Ind4Rate', 'Ind4Points',
       'Ind4PossiblePoints', 'Ind4HNRate', 'Ind4HNPoints',
       'Ind4HNPossiblePoints', 'Ind5Rate', 'Ind5Points', 'Ind5PossibleP
oints',
       'Ind6Rate', 'Ind6Points', 'Ind6PossiblePoints', 'Ind7Rate',
       'Ind7points', 'Ind7PointsPossible', 'Ind8Rate', 'Ind8Points',
       'Ind8PointsPossible', 'Ind9Rate', 'Ind9Points', 'Ind9PointsPossi
ble',
       'GradGapFlag', 'Ind10Rate', 'Ind10Points', 'Ind10PossiblePoint
s',
       'Ind11FitnessRate', 'Ind11ParticipationRate', 'Ind11Points',
       'Ind11PossiblePoints', 'Ind12Rate', 'Ind12Points',
       'Ind12PossiblePoints', 'Grad6YrRateHN', 'Grad6YrRateNHN',
       'Grad6YrHNNHNDiff', 'Threshold_Mean_ELA', 'Threshold_Mean_Math',
       'Threshold_Mean_Sci', 'Threshold_Mean_Grad', 'Classification',
       'State Category', '# of Distinctions',
       'School of Distinction - Highest Performing Overall',
       'School of Distintion - Highest Performing High Needs'],
      dtype='object')
```

By looking at the names of the columns, it seems that `Total Points` could be a relevant statistic, let's see if there is a linear relationship between `Total Points` and `Grad6YrRateHN` by plotting a scatter plot.

In [9]:
```
plt.scatter(Hartford_data['Grad6YrRateHN'], Hartford_data['Total Points'
])

plt.title('Graduation Rate vs. Total Points')
plt.xlabel('Graduation Rate')
plt.ylabel('Total Points');
```



Awesome! There looks like there is a linear relationship between `Total Points` and `Grad6YrRateHN`. As graduation rate increases, so does total points.

Let's make a DataFrame where our columns only contain these two statistics.

In [10]:
```
cleaned_data = Hartford_data[['Total Points', 'Grad6YrRateHN']]

cleaned_data.head()
```

Out[10]:

	Total Points	Grad6YrRateHN
9	1129.3	93.2
14	1077.2	93.2
18	1048.0	96.8
23	903.3	98.3
91	962.1	82.7

Now that we have our X and Y variables, we can fit a linear regression model to it! Let's use Python to split our data up into train and test sets.

In [11]:
```
train_x = np.array(cleaned_data['Total Points'][:38])
test_x = np.array(cleaned_data['Total Points'][38:])

train_y = np.array(cleaned_data['Grad6YrRateHN'][:38])
test_y = np.array(cleaned_data['Grad6YrRateHN'][38:])
```

Awesome. Now that we have our train and test sets, let's create our TensorFlow model. This TensorFlow Regression model was adapted from (https://www.geeksforgeeks.org/linear-regression-using-tensorflow/ (https://www.geeksforgeeks.org/linear-regression-using-tensorflow/)). First we will define `X` and `Y` with TensorFlow Placeholders. We will also create the variable `n` which we will use later for our cost function.

NOTE: We are dividing our x-values by 50 so we can save memory.

In [12]:
```
x = train_x / 50
y = train_y

n = len(x)

X = tf.placeholder("float")
Y = tf.placeholder("float")
```

Great! Now, let's initialize our weight and bias variables as `W` and `b`. Also, we will define our variables for the learning rate of our model, and the number of iterations it will use to calculate our regression line.

In [21]:
```
W = tf.Variable(np.random.randn(), name = "W")
b = tf.Variable(np.random.randn(), name = "b")

learning_rate = 0.01
training_epochs = 1000
```

We will set `y_pred` to the `y = MX + B` function for a line, except with our weight and bias variables substituted in.

In [28]:
```
y_pred = tf.add(tf.multiply(X, W), b)
```

Now we will define our cost function as `cost` and our optimization technique as `optimizer`. We will be using the mean squared error cost function with gradient descent which minimizes loss. For more info on cost functions and gradient descent and how they are implemented in Machine Learning models, check this out: https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220 (https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220)

In [30]:
```
cost = tf.reduce_sum(tf.pow(y_pred-Y, 2)) / (2 * n)

optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(co
st)
```

Now it's time to run our model! Run the cell below to calculate our `training_cost`, `weight`, and `bias`.

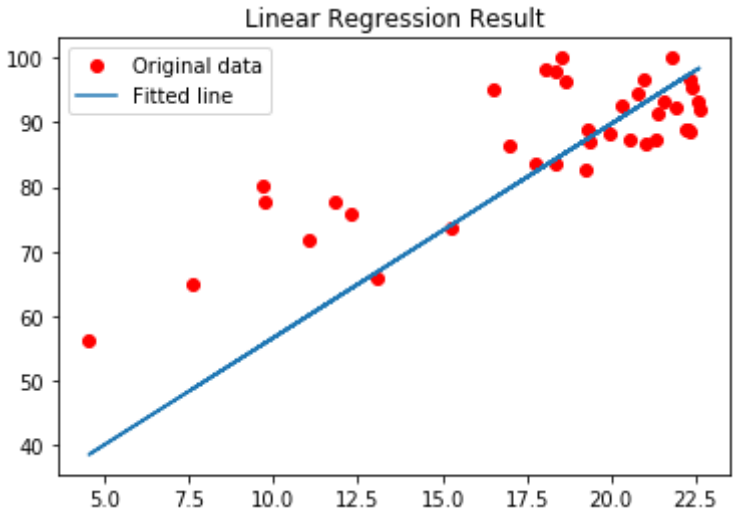
NOTE: Using code adapted from (https://www.geeksforgeeks.org/linear-regression-using-tensorflow/ (https://www.geeksforgeeks.org/linear-regression-using-tensorflow/))

In [32]:
```
# Starting the Tensorflow Session
with tf.Session() as sess:

    # Initializing the Variables
    sess.run(tf.global_variables_initializer())

    # Iterating through all the epochs
    for epoch in range(training_epochs):

        # Feeding each data point into the optimizer using Feed Dictiona
ry
        for (_x, _y) in zip(x, y):
            sess.run(optimizer, feed_dict = {X : _x, Y : _y})

        # Displaying the result after every 50 epochs
        if (epoch + 1) % 50 == 0:
            # Calculating the cost at every epoch
            c = sess.run(cost, feed_dict = {X : x, Y : y})
            print("Epoch", (epoch + 1), ": cost =", c, "W =", sess.run(W
), "b =", sess.run(b))

    # Storing necessary values to be used outside the Session
    training_cost = sess.run(cost, feed_dict ={X: x, Y: y})
    weight = sess.run(W)
    bias = sess.run(b)
```

```
Epoch 50 : cost = 109.96796 W = 4.446859 b = 1.1219581
Epoch 100 : cost = 104.77552 W = 4.369138 b = 2.6570752
Epoch 150 : cost = 99.88193 W = 4.293745 b = 4.146205
Epoch 200 : cost = 95.2698 W = 4.2206106 b = 5.590731
Epoch 250 : cost = 90.92277 W = 4.1496663 b = 6.991983
Epoch 300 : cost = 86.82538 W = 4.0808477 b = 8.351258
Epoch 350 : cost = 82.96305 W = 4.0140905 b = 9.669821
Epoch 400 : cost = 79.32216 W = 3.9493327 b = 10.948893
Epoch 450 : cost = 75.889854 W = 3.886515 b = 12.189645
Epoch 500 : cost = 72.654045 W = 3.8255796 b = 13.393212
Epoch 550 : cost = 69.60325 W = 3.7664692 b = 14.560734
Epoch 600 : cost = 66.7267 W = 3.7091286 b = 15.693299
Epoch 650 : cost = 64.014366 W = 3.6535072 b = 16.791914
Epoch 700 : cost = 61.45671 W = 3.5995514 b = 17.857624
Epoch 750 : cost = 59.044735 W = 3.547212 b = 18.891413
Epoch 800 : cost = 56.77001 W = 3.496441 b = 19.894226
Epoch 850 : cost = 54.624607 W = 3.447191 b = 20.866993
Epoch 900 : cost = 52.600945 W = 3.3994136 b = 21.810669
Epoch 950 : cost = 50.692146 W = 3.35307 b = 22.726027
Epoch 1000 : cost = 48.891415 W = 3.308113 b = 23.614
```

We will make our `predictions` using the `y = MX + B` function for a line and will plot it!

In [33]:
```
predictions = weight * x + bias

plt.plot(x, y, 'ro', label ='Original data')
plt.plot(x, predictions, label ='Fitted line')
plt.title('Linear Regression Result')
plt.legend()
plt.show()
```



Awesome! Our regression line looks good. Now that we have a trained model, let's use it to predict `Grad6YrRateHN` from `TotalPoints` on our test data.

In [39]:
```
x = test_x / 50

predictions_test = weight * x + bias
```

Let's look at our predictions for `Grad6YrRateHN` .

In [40]:
```
print(predictions_test)
```

```
[90.0342951  92.2308822  88.84999062 96.31970999 93.80554404 94.16943648
 92.15148749 85.51541261 85.85945638 80.057026  ]
```

Now let's look at our actual Y values and see if their similar!

In [42]:
```
print(test_y)
```

```
[93.3 76.2 76.9 90.  92.  89.1 85.2 87.2 83.6 84.8]
```

Awesome! Our predictions look pretty good, but if we wanted to be even more accurate, we would need more data. For now, this looks good!

CONCLUSION

What did we learn?

- What linear regression is
- How to define a linear regression model
- How to make predictions using linear regression
- Exploratory Data Analysis
- Using Python for Exploratory Data Analysis

In [ ]: