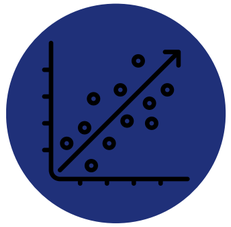
LIEANR REGRESSION

Regression Reading

REGRESSION

ទីវិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE



In this section we'll cover the Linear **regression**, including key modeling terminology, the types & goals of linear regression

TOPICS WE'LL COVER:

- **Linear Regression**
- **Regression with Python**
- **Asssumtion**
- **Optimization**

GOALS FOR THIS SECTION:

- Review the Regression Model We Learned From Statistics
- Build a Regression Model With a Pipeline
- Check Assumptions After Building the Model
- Minimize the prediction error using optimization techniques.

Linear Regression

ទិន្នាស្ថានសាន់ប៊ុយស៍
SUNRISE INSTITUTE

linear regression is a statistical model which estimates the linear relationship between a dependent variable and one or more explanatory variables or independent variable).

- **LinearRegression**
- **Linear with Python**
- **Assumtions**
- **Optimization**

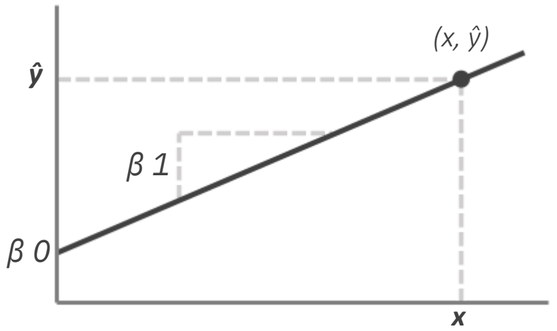
*The **predicted** value for the target*

The value for the feature

$$y = \beta_0 + \beta_1 x$$

*The **y-intercept***

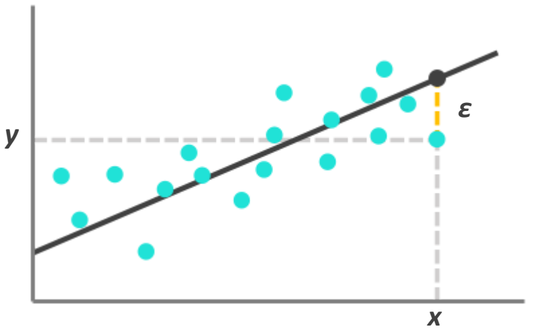
*The **slope** of the relationship*



*The **actual** value for the target*

$$y = \beta_0 + \beta_1 x + \epsilon$$

*The **error**, or **residual**, caused by the difference between the actual and predicted values*



Linear Regression

Multiple linear regression models use *multiple features* to predict the target

- In other words, it's the same linear regression model, but with additional "x" variables

SIMPLE LINEAR REGRESSION MODEL:

$$y_i = \beta_0 + \beta_1 x_{i1}$$

MULTIPLE LINEAR REGRESSION MODEL:

$$y_i = \underbrace{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon}$$

*Instead of just one "x", we have a **whole set of features** (and associated coefficients) to help predict the target (y)*

GOALS OF Linear REGRESSION

វិទ្យាស្ថានសាន់ប៊ុយស៍
SUNRISE INSTITUTE

Linear Regression models are used for two primary goals: **prediction** and **inference**

LinearRegression

Linear with Python

Optimization

Implementation

The goal shapes the modeling approach, including the regression algorithm used, the complexity of the model, and more

PREDICTION

- Used to **predict** the target as accurately as possible
- *“What is the predict charges for a client given their age?”*

INFERENCE

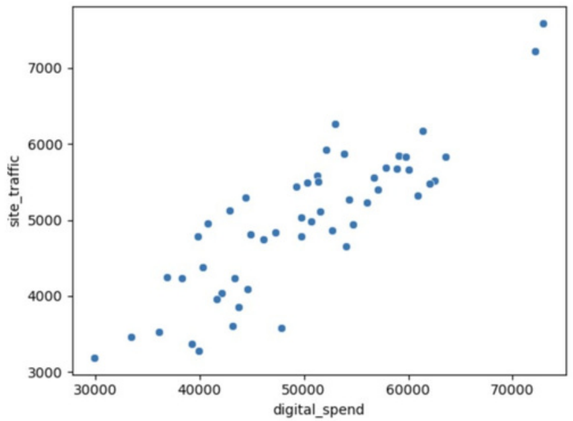
- Used to **understand the relationships** between the features and target
- *“How much do a age impact its charges?”*

LINEAR REGRESSION

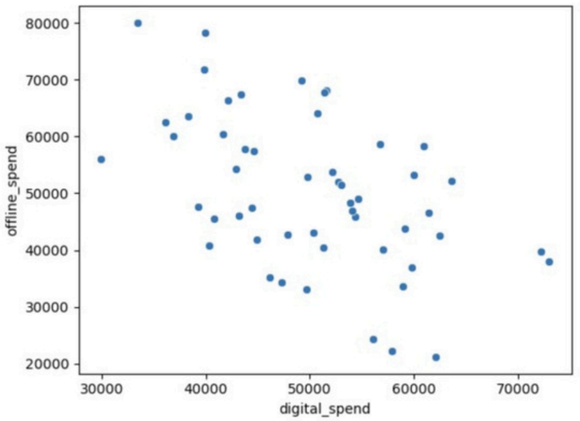
The **correlation coefficient** ‘r’ is single number that shows the linear relationship between two variables.

- The correlation coefficient varies between -1 and +1.
- -1 means a perfect inverse relationship,
- 0 means no relationship, and +1 means a perfect

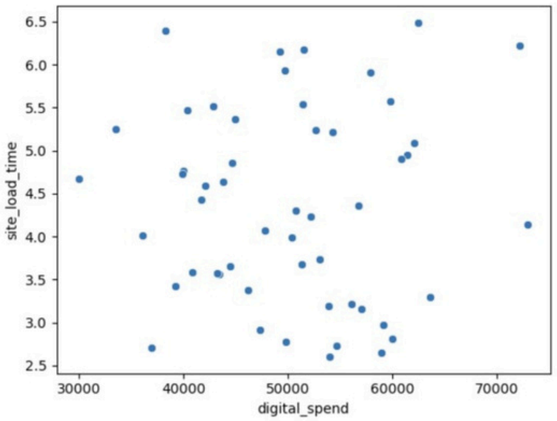
r = 0.858

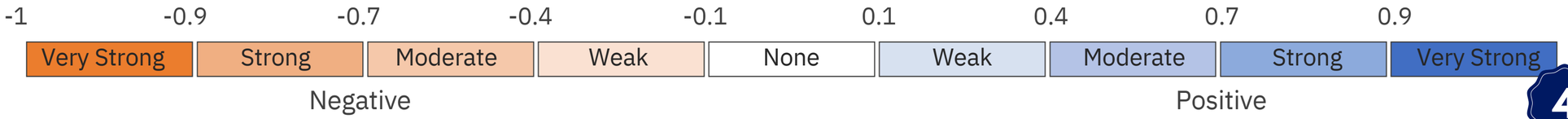


r= -0.499



r= 0.008





Regression Metrics

Model evaluation (Metric) -Regression

Some common regression metrics in scikit-learn

with examples

- **Mean Absolute Error (MAE)**
- **Mean Squared Error (MSE)**
- **R-squared (R^2) Score**
- **Root Mean Squared Error (RMSE)**

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Model evaluation (Metric) -Regression

$$y_i \text{ (true values)} = [2.5, 3.7, 1.8, 4.0, 5.2]$$

$$\hat{y}_i \text{ (predicted values)} = [2.1, 3.9, 1.7, 3.8, 5.0]$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

IMPLEMENTATION

These Python libraries are used fit regression models: **statsmodels** & **scikit-learn**

- **Regression**
- **Linear with Python**
- **Optimization**
- **Implementation**

statsmodels

- **Ideal if your goal is inference**
- Similar output to other tools (SAS, R, Excel)
- Easy access to dozens of statistical tests
- Harder to leverage in production ML

learn

- **Ideal if your goal is prediction**
- Most popular ML library in Python
- Has various models for easy
- comparison Designed to be deployed to production

Both libraries **use the same math** and return the same regression equation! We will begin by focusing on statsmodels, but once we have the fundamentals of regression down, we'll introduce scikit-learn

REGRESSION IN STATSMODELS

You can fit a **regression in statsmodels** with just a few lines of code:

```
import statsmodels.api as sm

x_train_con = sm.add_constant(x_train)
model = sm.OLS(y_train,x_train_con).fit()
model.summary()
```

*1) Import **statsmodels.api** (standard alias is **sm**)*

*2) Create an "X" DataFrame with your **feature(s)** and **add a constant***

*3) Create a "y" DataFrame with your **target***

*4) Call **sm.OLS(y, X)** to set up the model, then use **.fit()** to build the model*

*5) Call **.summary()** on the model to review the model output*

Why do we need to add a constant?

- Statsmodels assumes you want to fit a model with a line that runs through the origin (0, 0)
- sm.add_constant() lets statsmodels calculate a y-intercept other than 0 for the model
- Most regression software *(like sklearn)* takes care of this step behind the scenes

REGRESSION IN STATSMODELS

You can fit a **regression in statsmodels** with just a few lines of code:

Regression

Linear with Python

Optimization

Implementation

```
import statsmodels.api as sm

x_train_con = sm.add_constant(x_train)
model = sm.OLS(y_train,x_train_con).fit()
model.summary()
```

The model output can be intimidating the first time you see it, but we'll cover the important pieces in the next few lessons and later sections!

OLS Regression Results

Dep. Variable:	charges	**R-squared:**	0.918
Model:	OLS	**Adj. R-squared:**	0.918
Method:	Least Squares	**F-statistic:**	9714.
Date:	Sat, 30 Mar 2024	**Prob (F-statistic):**	0.00
Time:	22:43:12	**Log-Likelihood:**	-7288.4
No. Observations:	864	**AIC:**	1.458e+04
Df Residuals:	862	**BIC:**	1.459e+04
Df Model:	1		
Covariance Type:	nonrobust		

Model summary statistics

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-3400.3622	112.847	-30.133	0.000	-3621.849	-3178.876
age	266.8456	2.707	98.558	0.000	261.532	272.160

Variable summary statistics

Omnibus:	707.070	**Durbin-Watson:**	1.973
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	24364.367
Skew:	3.460	**Prob(JB):**	0.00
Kurtosis:	28.078	**Cond. No.**	124.

Residual (error) statistics

INTERPRETING THE MODEL

ទីវិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

To **interpret the model**, use the “coef” column in the variable summary statistics

- Regression
- **Linear with Python**
- Optimization
- Implementation

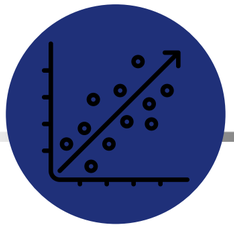
```
import statsmodels.api as sm

x_train_con = sm.add_constant(x_train)
model = sm.OLS(y_train,x_train_con).fit()
model.summary()
```

	coef	std err	t	P>|t|	[0.025	0.975]
const	-3400.3622	112.847	-30.133	0.000	-3621.849	-3178.876
age	266.8456	2.707	98.558	0.000	261.532	272.160

$\hat{y} = -3400 + 266.8456x$

How do we interpret this?

- Technically, Intercept (-3400): When x (the age of the client) is 0, the predicted outcome y_pre is -3400.



R-SQUARED

Regression

Linear with Python

Optimization

Implementation

R-squared, or coefficient of determination, measures how much better the model is at predicting the target than using its mean *(our best guess without using features)*

- R-squared values are bounded between 0 and 1 on training data

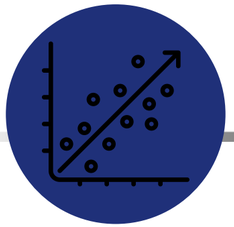
```
import statsmodels.api as sm

x_train_con = sm.add_constant(x_train)
model = sm.OLS(y_train,x_train_con).fit()
model.summary()
```

***Squaring the correlation** between the feature and target yields R2 in simple linear regression (this doesn't hold in multiple linear regression)*

OLS Regression Results

Dep. Variable:	charges	**R-squared:**	0.918
Model:	OLS	**Adj. R-squared:**	0.918
Method:	Least Squares	**F-statistic:**	9714.
Date:	Sat, 30 Mar 2024	**Prob (F-statistic):**	0.00
Time:	22:43:12	**Log-Likelihood:**	-7288.4
No. Observations:	864	**AIC:**	1.458e+04
Df Residuals:	862	**BIC:**	1.459e+04
Df Model:	1		
Covariance Type:	nonrobust		

*The model explains 91**.9%** of the variation in price not explained by the mean of charges*



R-SQUARED

Regression

Linear with Python

Optimization

Implementation

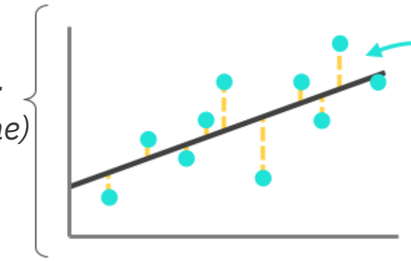
R-squared, or coefficient of determination, measures how much better the model is at predicting the target than using its mean *(our best guess without using features)*

R-squared values are bounded between 0 and 1 on training data

$$R2 = 1 - \frac{SSE}{SST}$$

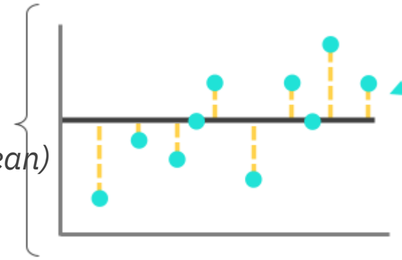
*This is the **sum of squared error** (distance between values & line)*

*This is the **sum of squared total** (distance between values & mean)*



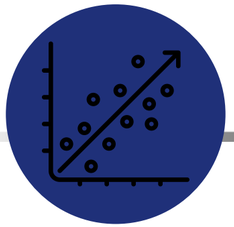
This is the variation of "y" not explained by "x"

- *Reducing error will increase R2*
- *A perfect model has an error of 0, or R2 of 1*



This is the variation of "y" around its mean

- *If R2 = 0, the model is no better than the mean of y*



R-SQUARED

Regression

Linear with Python

Optimization

Implementation

Adjusted R^2

A variation of the R^2 regression evaluation metric that penalizes unnecessary explanatory variables