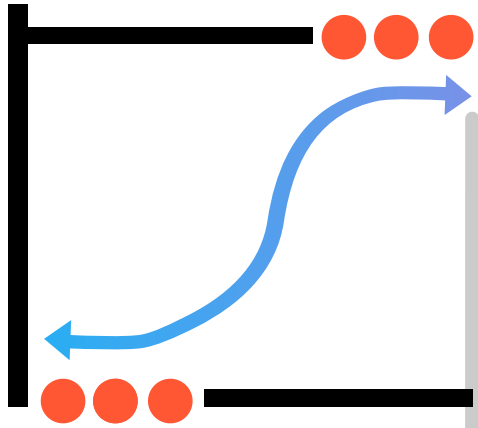
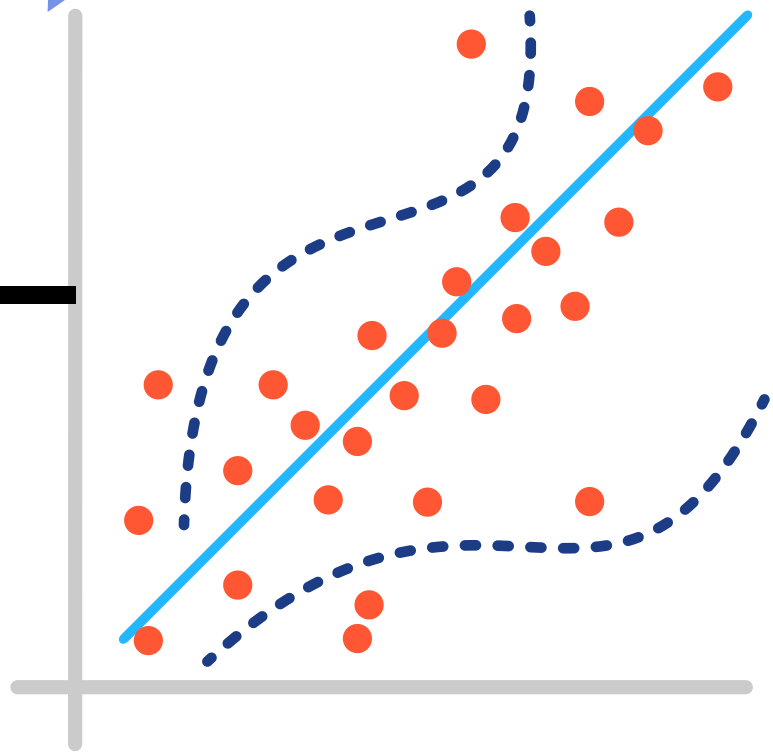
WELLCOME

REGRESSION ANALYSIS

EMBARKING ON A JOURNEY INTO DATA SCIENCE

YA MANON





ABOUT THIS SERIES

PART 1 → **PART 2** → **PART 3**

Data Prep & EDA | Regression | Classification

COURSE OUTLINE

1. **Simple Linear Regression**

   Build simple linear regression models in Python and learn about the metrics and statistical tests that help evaluate their quality and output

2. **Multiple Linear Regression**

   Build multiple linear regression models in Python and evaluate the model fit, perform variable selection, and compare models using error metrics

3. **Model Assumptions**

   Review the assumptions of linear regression models that need to be met to ensure that the model's predictions and interpretation are valid

4. **Model Testing & Validation**

   Test model performance by splitting data, tuning the model with the train & validation data, selecting the best model, and scoring it on the test data

5. **Feature Engineering**

   Apply feature engineering techniques for regression models, including dummy variables, interaction terms, binning, and more

WELCOME TO DATA SCIENCE CLASS

THE
OBJECTIVES

1. **Explore** & visualize the data
2. **Prepare** the data for modelling
3. **Apply regression algorithms** to the data
4. **Evaluate** how well your models fit
5. **Select** the best model and interpret it

DATA SCIENCE

DATA SCIENCE WORKFLOW

- What is Data Science
- Essential Skills
- Machine Learning
- Data Science Workflow

The **data science workflow** consists of scoping the project, gathering, cleaning and exploring the data, applying models, and sharing insights with end users

This is not a linear process! *You'll likely go back to further gather, clean and explore your data*

REGRESSION

REGRESSION

In this section we'll cover the basics of **regression**, including key modeling terminology, the types & goals of regression analysis, and the regression modeling workflow

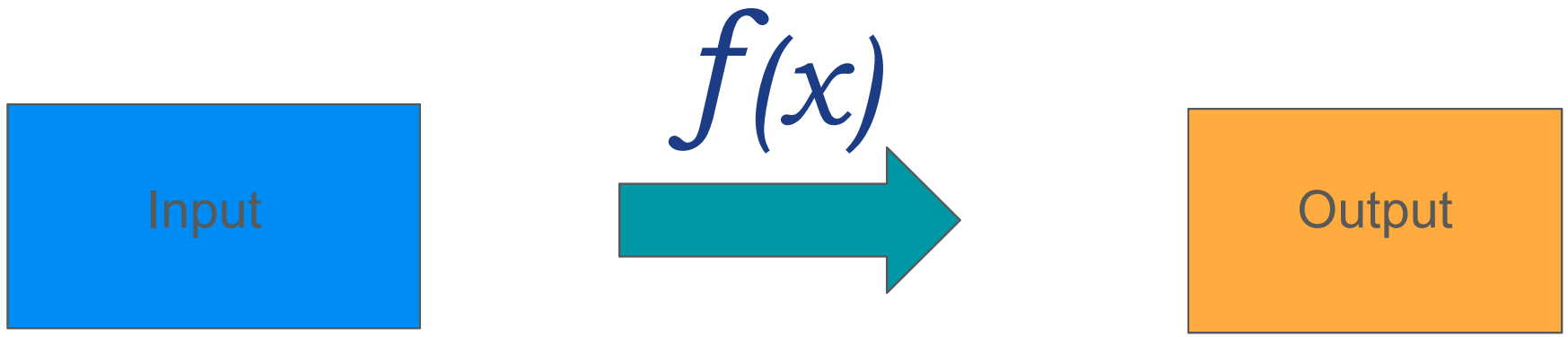
TOPICS WE'LL COVER:

- Regression
- Types of Regression
- Goals of Regression
- The Model WorkFlow

GOALS FOR THIS SECTION:

- Introduce the basics of regression modeling
- Understand key modeling terminology
- Discuss the different goals of regression modeling
- Review the regression modeling workflow

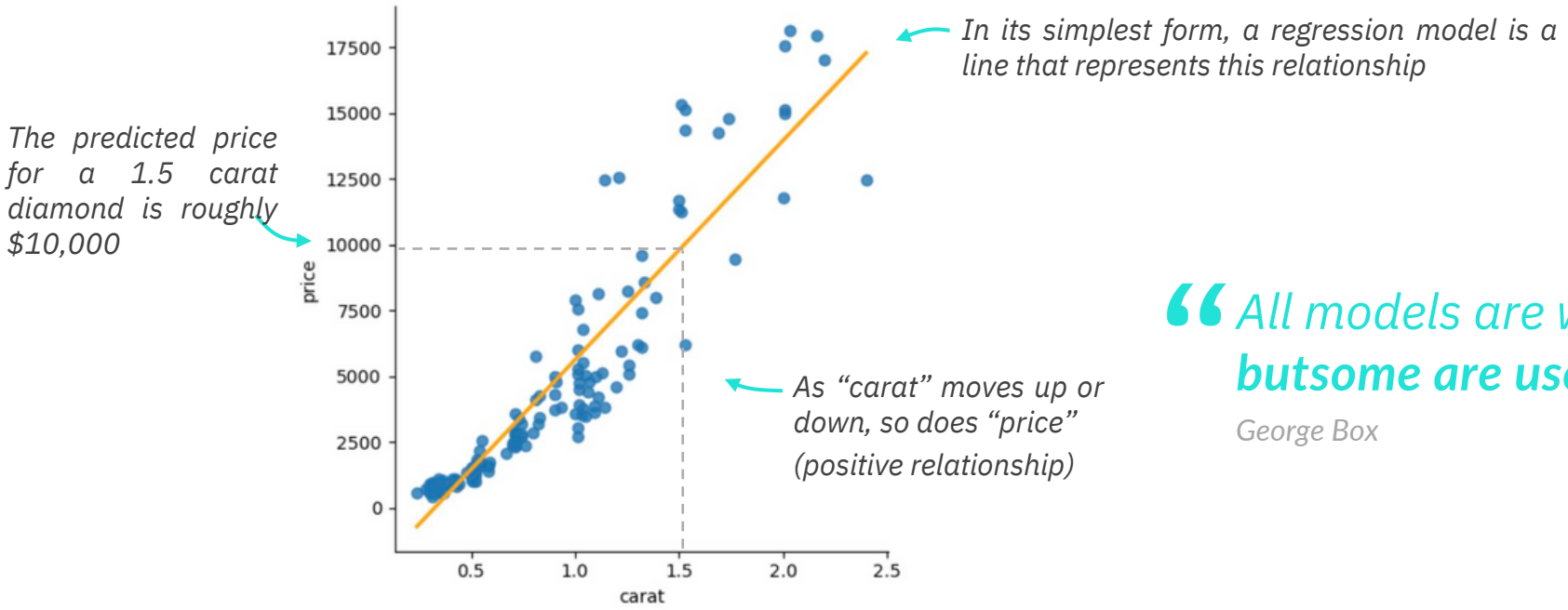
Model?



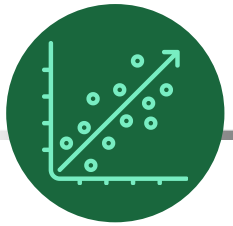
REGRESSION

Regression analysis is statistical model technique used to predict a numeric variable (target) by modeling its relationship with a set of other variables (features)

EXAMPLE *Predicting the price of diamonds based on carat weight*



> *All models are wrong,* ***butsome are useful***
>
> George Box



REGRESSION

Regression

Goals of Regression

Types of Regression

The Model Workflow

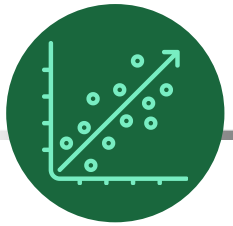
Regression analysis is statistical model technique used to predict a numeric variable (target) by modeling its relationship with a set of other variables (features)

y Target

- This is the variable **you're trying to predict**
- The target is also known as *"Y"*, *"model output"*, *"response"*, or *"dependent"* variable
- Regression helps understand how the target variable is impacted by the features

x Features

- These are the variables that **help you predict the target variable**
- Features are also known as *"X"*, *"model inputs"*, *"predictors"*, or *"independent"*
- variables Regression helps understand how the features impact, or *predict*, the target



REGRESSION

Regression

Goals of Regression

Types of Regression

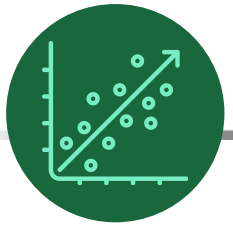
The Model Workflow

EXAMPLE *Predicting the price of diamonds based on diamond characteristics*

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

charges *is our* ***target****, since it's what we want to predict*
Since price is numerical, we'll

use regression to predict it



REGRESSION

- **Regression**
- **Goals of Regression**
- **Types of Regression**
- **The Model Workflow**

EXAMPLE *Predicting the price of diamonds based on diamond characteristics*

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
5	31	female	25.740	0	no	southeast	**???**

*We'll use records with **observed values** for both the features and target to "train" our regression model...*

*...then apply that model to new, **unobserved values** containing features but no target*

This is what our model will predict!

GOALS OF REGRESSION

Regression

Goals of Regression

Types of Regression

The Model Workflow

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

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

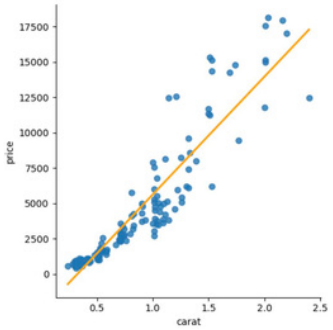
- Used to **understand the relationships** between the features and target
- *"How much do a age impact its charges?"*

TYPES OF REGRESSION

- Regression
- Goals of Regression
- **Types of Regression**
- The Model Workflow

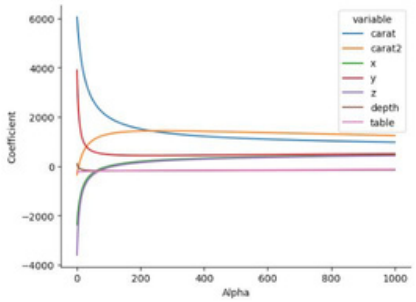
These are some of the major **types of regression** modeling techniques:

Linear Regression



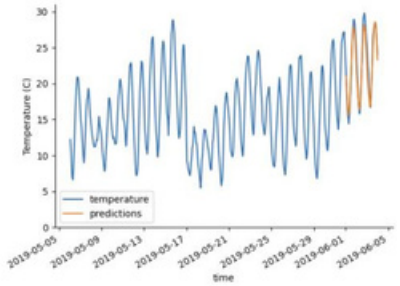
Models the relationship using between the features & target a linear equation

Regularized Regression



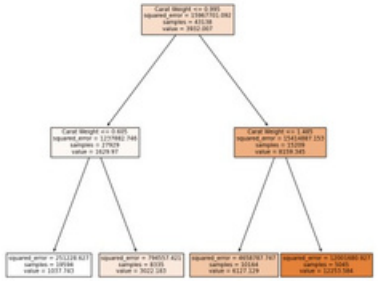
An extension of linear regression that penalizes model complexity

Time-Series Forecasting



Predicts future data using historical trends & seasonality

Tree-Based Regression

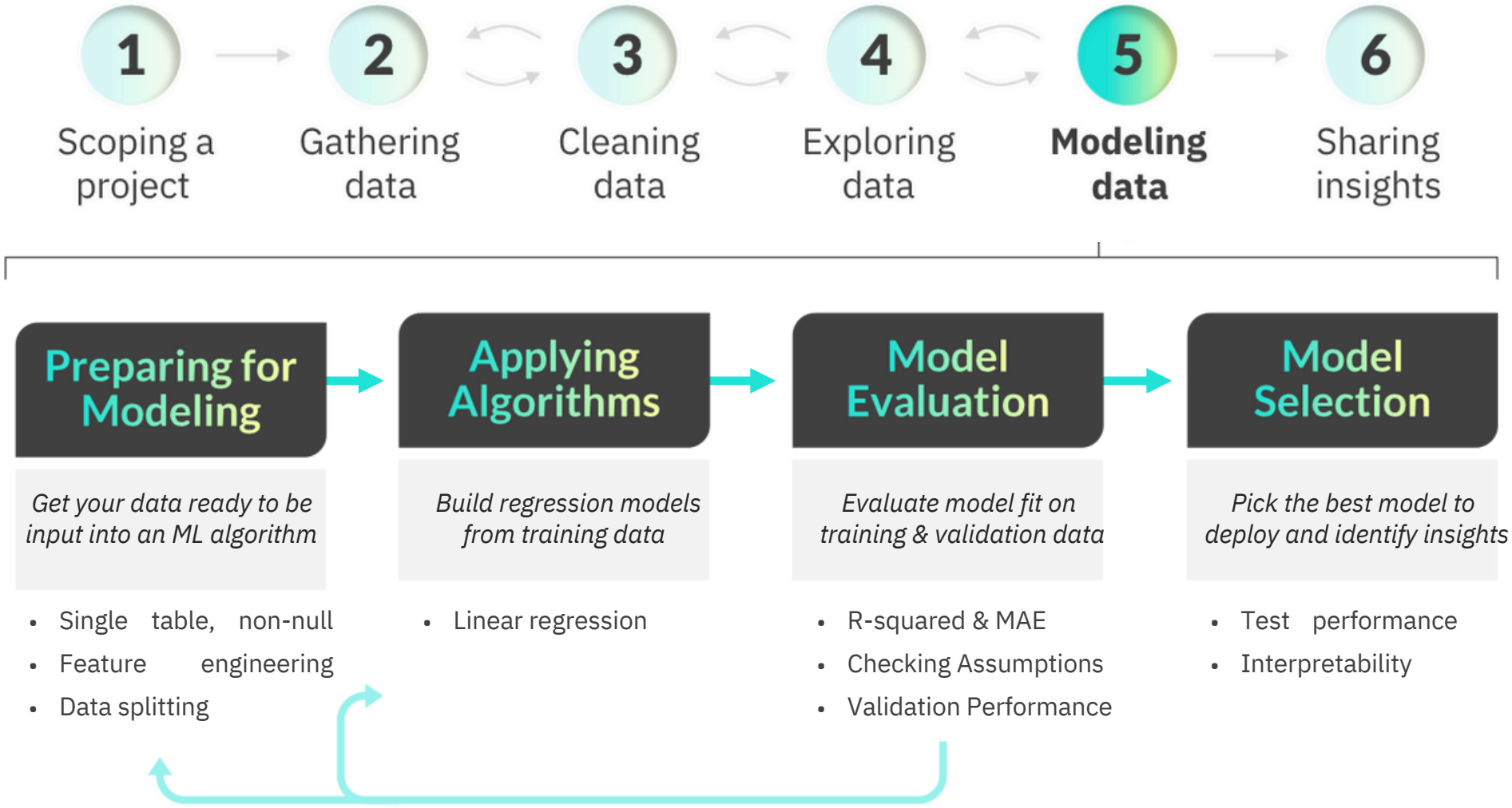


Splits data by maximizing the difference between groups

Even though **logistic regression** (which you may have heard of) has "regression" in its name, it's actually a classification modeling technique!

REGRESSION MODELING WORKFLOW

- Regression
- Goals of Regression
- Types of Regression
- The Model Workflow



KEY TAKEAWAYS

- Regression modeling is used to **predict numeric values**
  - *There are several types of regression models, but we will mostly focus on linear regression in this course*
- The **target** is the value we want to predict, and the **features** help us predict it
  - *The target is also known as "Y", "model output", "response", or "dependent" variable*
  - *Features are also known as "X", "model inputs", "predictors", or "independent" variables*
- Regression Modeling has two primary goals: **prediction** and **inference**
  - *Inference is used to understand the relationship between the features and target*
  - *Prediction focuses on predicting the target as accurately as possible*
- The **modeling workflow** is designed to ensure strong performance
  - *Splitting data, feature engineering, and model validation all work to ensure your model is as accurate as possible*

SIMPLE LINEAR REGRESSION

LINEAR REGRESSION MODEL

- **Linear Regression Model**
- **Least Squared Error**
- **Regression in Python**
- **Making Prediction**
- **Evaluation Metrics**

The **linear regression model** is an equation that best describes a linear relationship.

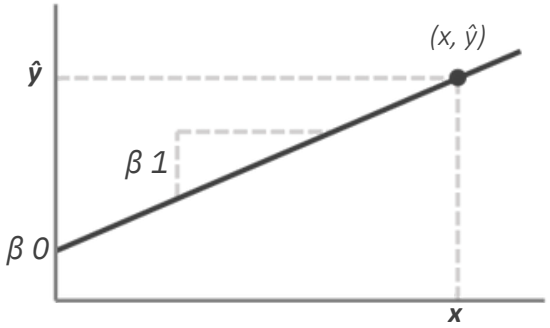
*The **predicted** value for the target*

The value for the feature

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

*The **y-intercept***

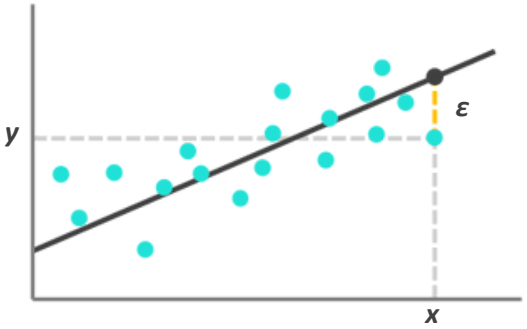
*The **slope** of the relationship*

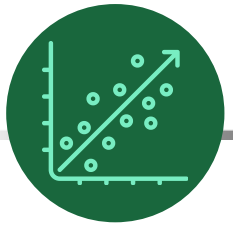


*The **actual** value for the target*

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

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





LINEAR REGRESSION MODEL

Linear Regression Model

Least Squared Error

Regression in Python

Making Prediction

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

LINEAR REGRESSION MODEL

Linear Regression Model

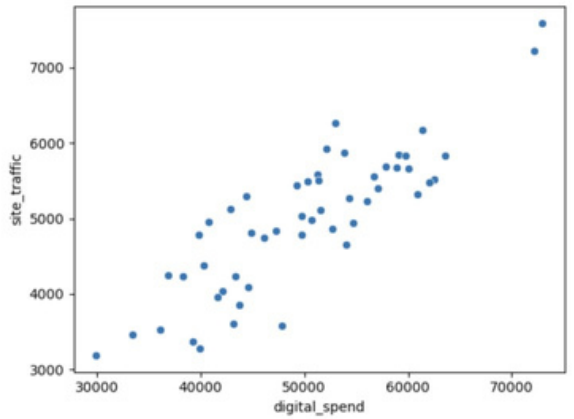
Least Squared Error

Regression in Python

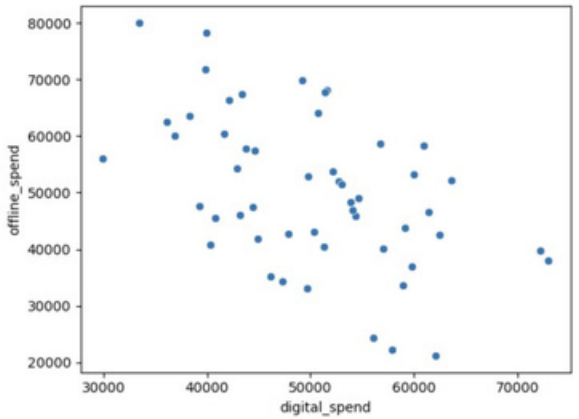
Making Prediction

You can use the **.corr()** method to calculate correlations in Pandas
– df[“col1”].corr(df[“col2”])

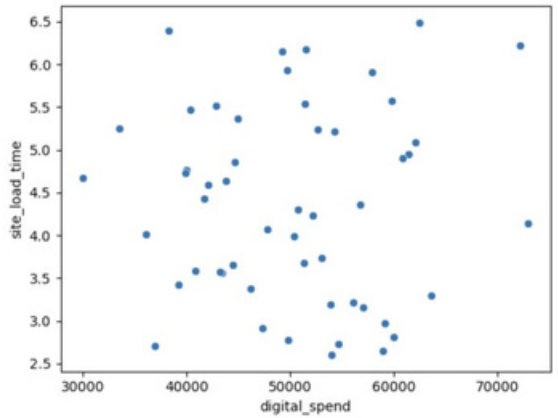
r = 0.858



r= -0.499



r= 0.008



-1	-0.9	-0.7	-0.4	-0.1	0.1	0.4	0.7	0.9	1
Very Strong	Strong	Moderate	Weak	None	Weak	Moderate	Strong	Very Strong	

Negative Positive

LINEAR REGRESSION MODEL

Linear Regression Model

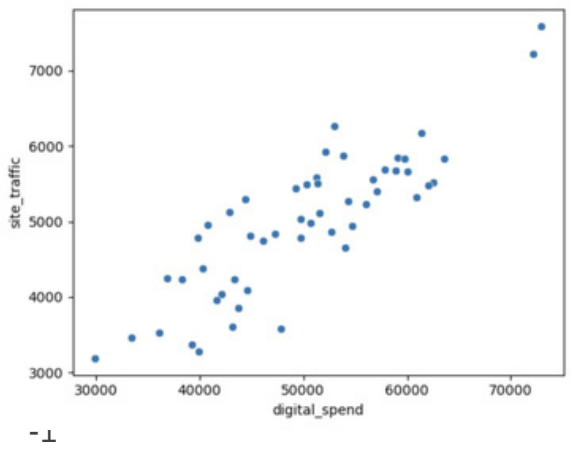
Least Squared Error

Regression in Python

Making Prediction

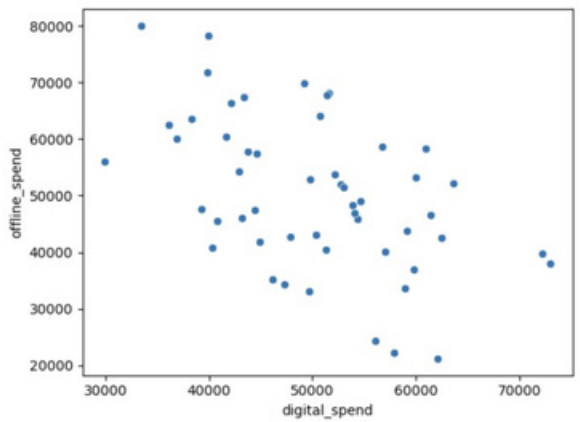
You can use the **.corr()** method to calculate correlations in Pandas – df[“col1”].corr(df[“col2”])

r = 0.858



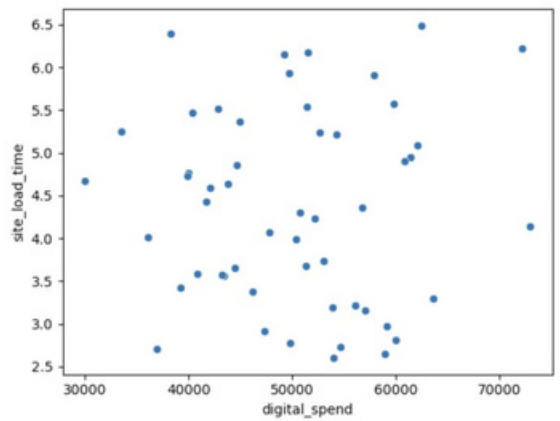
Strong positive correlation

r= -0.499

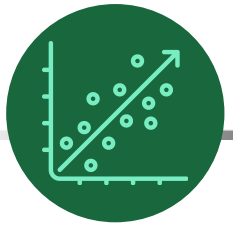


Moderate negative correlation

r= 0.008



No correlation



LEAST SQUARED ERROR

The **least squared error** method finds the line that best fits through the data

- It works by solving for the line that **minimizes the sum of squared error**
- The equation that minimizes error can be solved with **linear algebra**

Why "squared" error?

- Squaring the residuals converts them into **positive values**, and prevents positive and negative distances from cancelling each other out *(this makes the algebra to solve the line much easier, too)*
- One drawback of squared errors is that outliers can significantly impact the line *(more later!)*

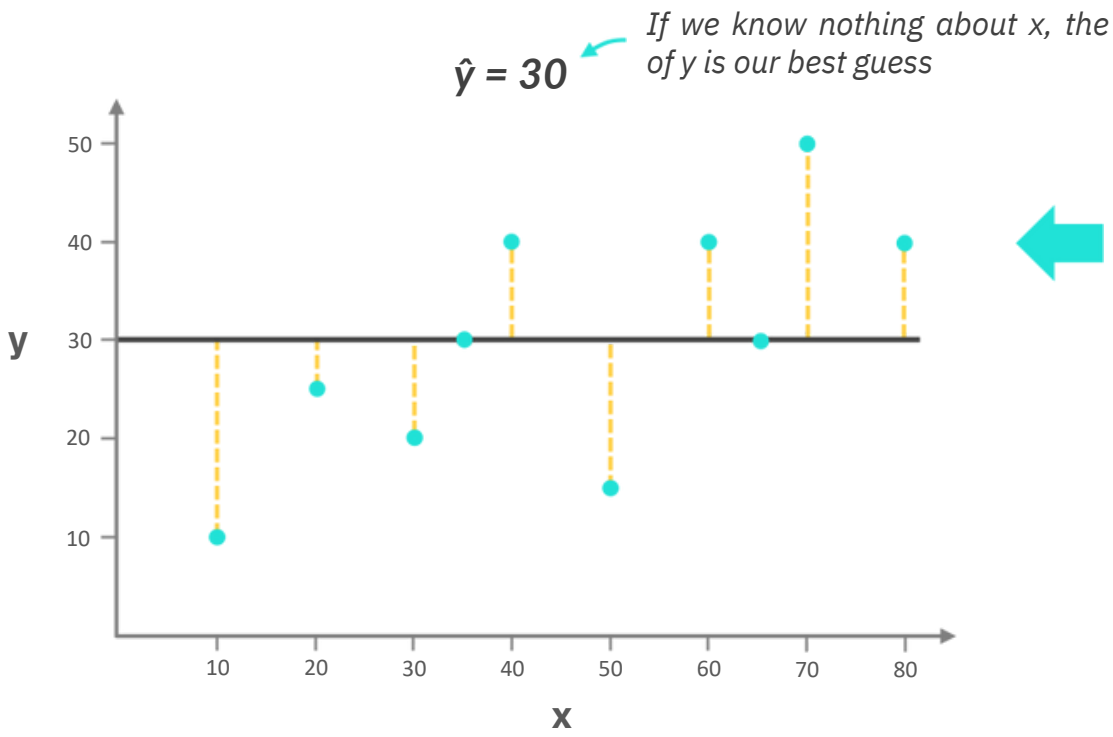
Ordinary Least Squares (OLS) is another term for traditional linear regression

There are other frameworks for linear regression that don't use least squared error, but they are rarely used outside of specialized domains

LEAST SQUARED ERROR

The **least squared error** method finds the line that best fits through the data

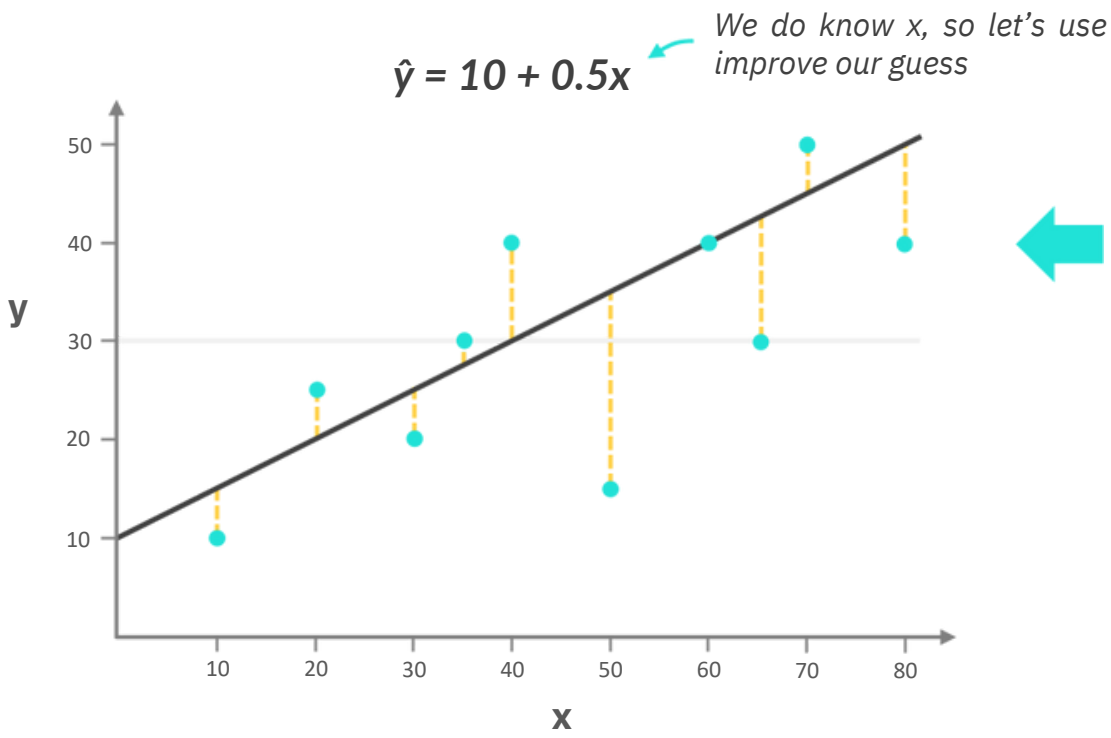
- It works by solving for the line that **minimizes the sum of squared error**
- The equation that minimizes error can be solved with **linear algebra**



LEAST SQUARED ERROR

The **least squared error** method finds the line that best fits through the data

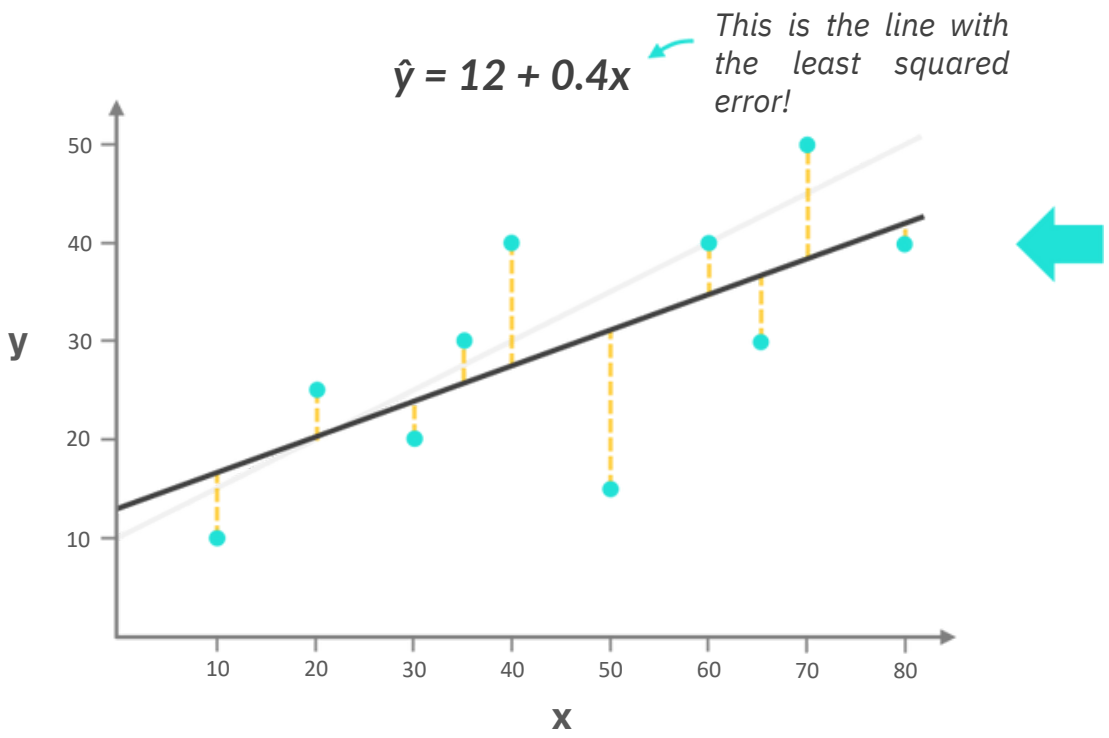
- It works by solving for the line that **minimizes the sum of squared error** The
- equation that minimizes error can be solved with **linear algebra**

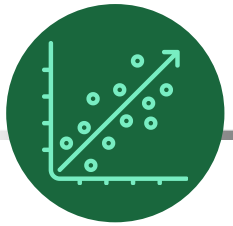


LEAST SQUARED ERROR

The **least squared error** method finds the line that best fits through the data

- It works by solving for the line that **minimizes the sum of squared**
- **error** The equation that minimizes error can be solved with **linear algebra**





REGRESSION IN PYTHON

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

statsmodels

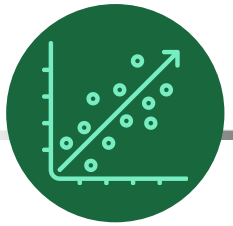
- **Ideal if your goal is inference**
- Similar output to other tools (SAS, R, Excel)
- Easy access to dozens of statistical tests
- Harder to leverage in production ML

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

- Linear Regression Model
- Least Squared Error
- **Regression in Python**
- Making Prediction
- Evaluation Metrics

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

- Linear Regression Model
- Least Squared Error
- **Regression in Python**
- Making Prediction
- Evaluation Metrics

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()
```

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

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!

INTERPRETING THE MODEL

- Linear Regression Model
- Least Squared Error
- Regression in Python
- Making Prediction
- Evaluation Metrics

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

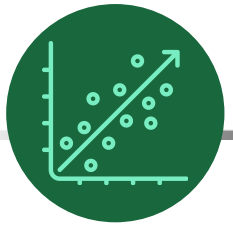
```
import statsmodels.api as sm

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.



MAKING PREDICTIONS

- Linear Regression Model
- Least Squared Error
- Regression in Python
- **Making Prediction**
- Evaluation Metrics

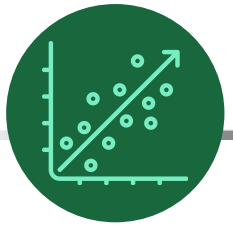
The **.predict()** method returns model predictions for single points or DataFrames

```
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

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

Age 45 predicted chnarges of $?



R-SQUARED

- Linear Regression Model
- Least Squared Error
- Regression in Python
- Making Prediction
- **Evaluation Metrics**

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

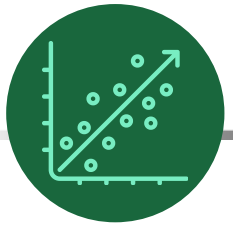
```
import statsmodels.api as sm

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*



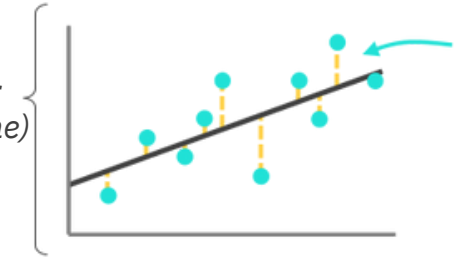
R-SQUARED

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

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

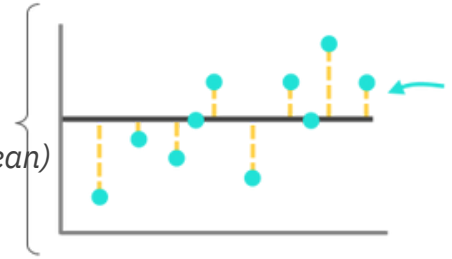
*This is the **sum of squared error** (distance between values & line)*



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

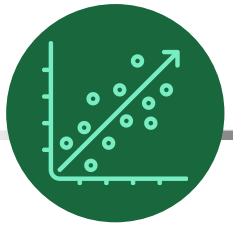
- *Reducing error will increase R2*
- *A perfect model has an error of 0, or R2 of 1*

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



This is the variation of "y" around its mean

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



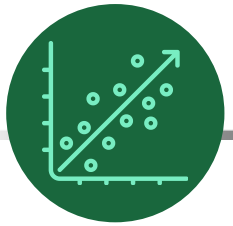
HYPOTHESIS TEST

Regression models include several **hypothesis tests**, including the F-test, that indicates whether our model is significantly better at predicting our target than using the mean of the target as the model

In other words, you're trying to find significant evidence that your model isn't useless

Steps for the hypothesis test:

1) State the **null** and **alternative hypotheses**
2) Set a **significance level** (α)
3) Calculate the **test statistic** and **pvalue**
4) Draw a **conclusion** from the test
    a) If $p \leq \alpha$, reject the null hypothesis *(you're confident the model isn't useless)*
    b) If $p > \alpha$, don't reject it *(the model is probably useless, and needs more training)*



HYPOTHESES & SIGNIFICANCE LEVEL

1)For F-Tests, the **null & alternative hypotheses** are always the same:

- **Ho: F=0** – The model has no special effect, meaning it's just as good as guessing the average.
- **Ha: F≠0** – The model does have a special effect,This means the model helps us make more accurate predictions compared to if we only used the average value as our guess.

*The hope is to **reject the null hypothesis***
(and therefore, accept the alternative)

2)The **significance level** is the threshold you set to determine when the evidence against your null hypothesis is considered "strong enough" to prove it wrong **α**

- This is set by **alpha ()**, which is the accepted probability of error
- The industry standard is **α= .05** *(this is what we'll use in the course)*

Some teams and industries set a much higher bar, such as .01 or even .001, making the null hypothesis **less likely to be rejected**

F-STATISTIC & P-VALUE

- Linear Regression Model
- Least Squared Error
- Regression in Python
- Making Prediction
- **Evaluation Metrics**

3)The **F-statistic** and associated **p-value** are part of the model summary and help understand the predictive power of the regression model *as a whole*

- The **F-statistic** is the ratio of variability the model explains vs the variability it doesn't
- The **p-value**, or F-significance, is the probability that your model predicts poorly

```
import statsmodels.api as sm

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

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 F-statistic is primarily used as a stepping-stone to calculate the p-value, which is **easier to interpret** and **more commonly used** in model diagnostics

HYPOTHESIS TEST CONCLUSION

- Linear Regression Model
- Least Squared Error
- Regression in Python
- Making Prediction
- **Evaluation Metrics**

4)Comparing the p-value and alpha lets us draw a **conclusion** from the test **α**

- **p≤** α reject the null hypothesis *(you're confident the model isn't useless)*
- **p>= α** don't reject it *(the model is probably useless, and needs more training)*

```
import statsmodels.api as sm

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

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		

Our F-statistic is much greater than 0, and the p-value is less than 0.05, so we can reject the null hypothesis (age is a good predictor of a charges price!)

T-STATISTICS & P-VALUES

- Linear Regression Model
- Least Squared Error
- Regression in Python
- Making Prediction
- Evaluation Metrics

The **T-statistics** and associated **p-values** are part of the model summary and help understand the predictive power of *individual model coefficients*

- It's essentially another hypothesis test designed to find which coefficients are useful

```
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

Since the p-value is lower than our alpha of 0.05, we can conclude that age is a good predictor of charges

(the constant also has a p-value lower than 0.05, but we can generally ignore insignificant p-values for the intercept term)

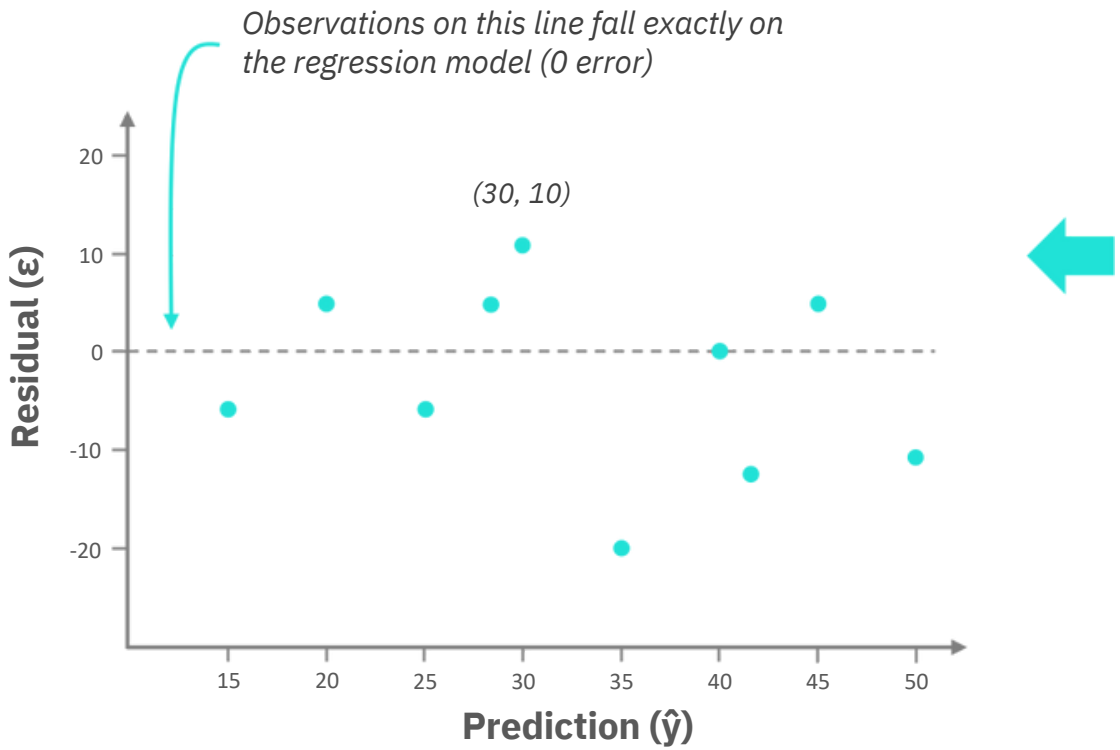
This will become more relevant when performing **variable selection** in multiple linear regression models (up next!)

RESIDUAL PLOTS

- Linear Regression Model
- Least Squared Error
- Regression in Python
- Making Prediction
- **Evaluation Metrics**

Residual plots show how well a model performs across the range of predictions

Ideally, residual plots should be normally distributed around 0



RESIDUAL PLOTS

- Linear Regression Model
- Least Squared Error
- Regression in Python
- Making Prediction
- **Evaluation Metrics**

Residual plots show how well a model performs across the range of predictions

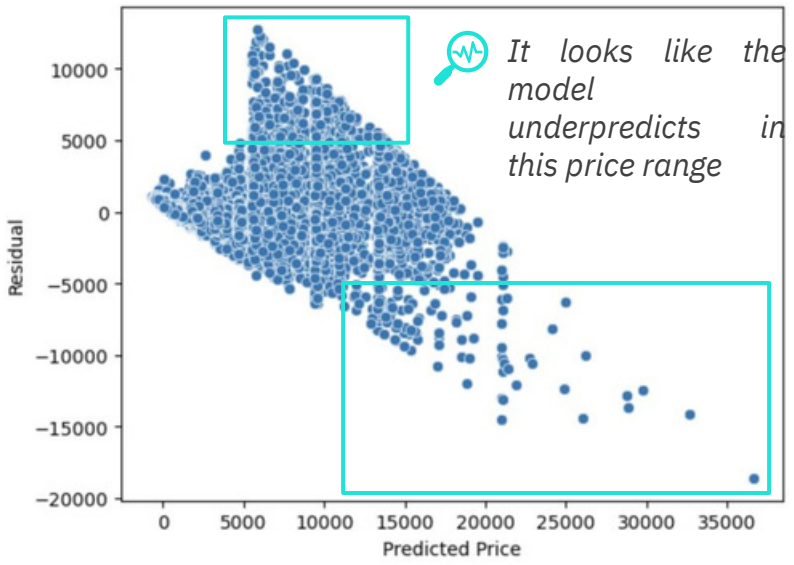
Ideally, residual plots should show errors to be normally distributed around 0

- **model.resid** returns a series with the residuals *(actual value - predicted value)*

```
residuals = pd.DataFrame(
    {
        "Carat": diamonds["carat"],
        "Price": diamonds["price"],
        "Predicted Price": model.predict(),
        "Residual": model.resid
    }
)

residuals.head()
```

	Carat	Price	Predicted Price	Residual
0	0.40	666	846.179416	-180.179416
1	0.77	1940	3716.060795	-1776.060795
2	0.58	2036	2242.337925	-206.337925
3	0.41	705	923.743778	-218.743778
4	0.81	3250	4026.318242	-776.318242

```
sns.scatterplot(residuals, x="Predicted Price", y="Residual");
```



It looks like the model underpredicts in this price range

It looks like the model overpredicts as price increases