REGRESSION 101

REGRESSION 101

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 101
- Goals of Regression
- Types of Regression
- The Modeling Workflow

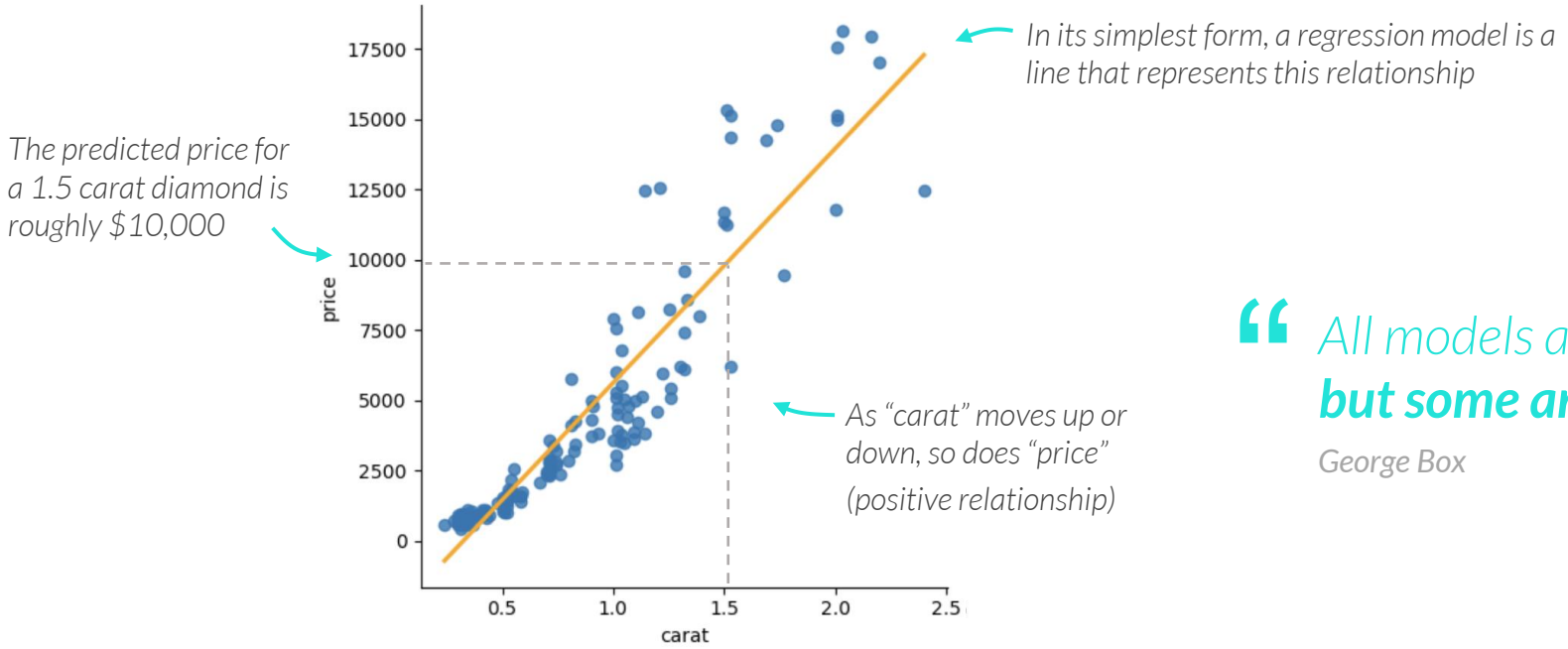
GOALS FOR THIS SECTION:

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

REGRESSION 101

Regression analysis is supervised learning 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,*
> ***but some are useful***
>
> *George Box*

REGRESSION 101

Regression analysis is supervised learning 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 101

EXAMPLE

Predicting the price of diamonds based on diamond characteristics

	carat	cut	color	clarity	depth	table	x	y	z	price
0	0.26	Ideal	H	IF	61.1	57.0	4.12	4.16	2.53	468
1	1.20	Ideal	H	VS1	61.6	57.0	6.82	6.85	4.21	8275
2	0.62	Good	E	SI2	59.4	65.0	5.54	5.48	3.27	1333
3	1.50	Good	I	VVS2	63.3	58.0	7.24	7.27	4.59	9909
4	1.27	Very Good	I	SI2	59.7	57.0	6.99	7.04	4.19	5371

***Price** is our **target**, since it's what we want to predict*

*Since price is **numerical**, we'll use regression to predict it*

***Carat**, **cut**, **color**, **clarity**, and **the rest of the columns** are all **features**, since they can help us explain, or predict, the price of diamonds*

REGRESSION 101

EXAMPLE

Predicting the price of diamonds based on diamond characteristics

	carat	cut	color	clarity	depth	table	x	y	z	price
0	0.26	Ideal	H	IF	61.1	57.0	4.12	4.16	2.53	468
1	1.20	Ideal	H	VS1	61.6	57.0	6.82	6.85	4.21	8275
2	0.62	Good	E	SI2	59.4	65.0	5.54	5.48	3.27	1333
3	1.50	Good	I	VVS2	63.3	58.0	7.24	7.27	4.59	9909
4	1.27	Very Good	I	SI2	59.7	57.0	6.99	7.04	4.19	5371
5	0.90	Ideal	E	SI2	62.1	56.0	6.25	6.19	3.86	???

*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 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 predicted price of a diamond given its characteristics?"*

INFERENCE

- Used to **understand the relationships** between the features and target
- *"How much do a diamond's size and weight impact its price?"*

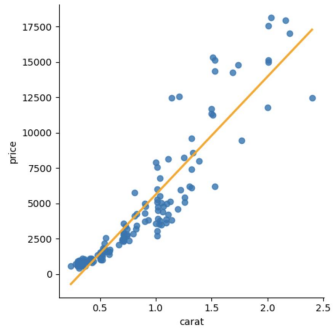
You often need to **strike a balance** between these goals – a model that is very inaccurate won't be too trustworthy for inference, and understanding the impact that variables have on predictions can help make them more accurate

TYPES OF REGRESSION

- Regression 101
- Goals of Regression
- Types of Regression
- The Modeling Workflow

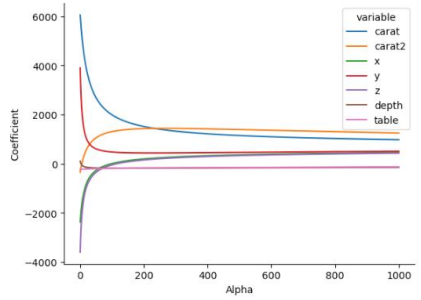
These are some of the major **types of regression** modeling techniques:

Linear Regression



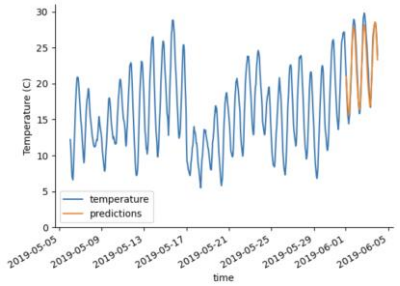
Models the relationship between the features & target using a linear equation

Regularized Regression



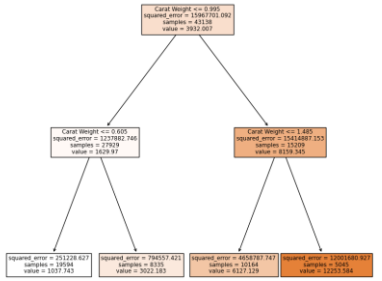
An extension of linear regression that penalizes model complexity

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 101
- Goals of Regression
- Types of Regression
- The Modeling Workflow

1 Scoping a project → 2 Gathering data ⇄ 3 Cleaning data ⇄ 4 Exploring data ⇄ **5 Modeling data** → 6 Sharing insights

Preparing for Modeling

Get your data ready to be input into an ML algorithm

- Single table, non-null
- Feature engineering
- Data splitting

Applying Algorithms

Build regression models from training data

- Linear regression
- Regularized regression
- Time series

Model Evaluation

Evaluate model fit on training & validation data

- R-squared & MAE
- Checking Assumptions
- Validation Performance

Model Selection

Pick the best model to deploy and identify insights

- Test performance
- Interpretability

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**

- *Prediction focuses on predicting the target as accurately as possible*
- *Inference is used to understand the relationship between the features and target*

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*

PRE-MODELING DATA PREP & EDA

PRE-MODELING DATA PREP & EDA

In this section we'll review the **data prep & EDA** steps required before applying regression algorithms, including key techniques to explore the target, features, and their relationships

TOPICS WE'LL COVER:

- EDA for Regression
- Exploring the Target
- Exploring the Features
- Linear Relationships
- Exploring Relationships
- Preparing for Modeling

GOALS FOR THIS SECTION:

- Visualize & explore the target and features
- Quantify the strength of linear relationships
- Identify relationships between the target and features, as well as between features themselves
- Review the data prep steps required for modeling

EDA FOR REGRESSION

Exploratory data analysis (EDA) is the process of exploring and visualizing data to find useful patterns & insights that help inform the modeling process

- In regression, EDA lets you identify & understand the most promising features for your model
- It also helps uncover potential issues with the features or target that need to be addressed

When performing **EDA for regression**, it's key to explore:

- The target variable
- The features
- Feature-target relationships
- Feature-feature relationships

Even though data cleaning comes before exploratory data analysis, it's common for EDA to **reveal additional data cleaning steps** needed before modeling

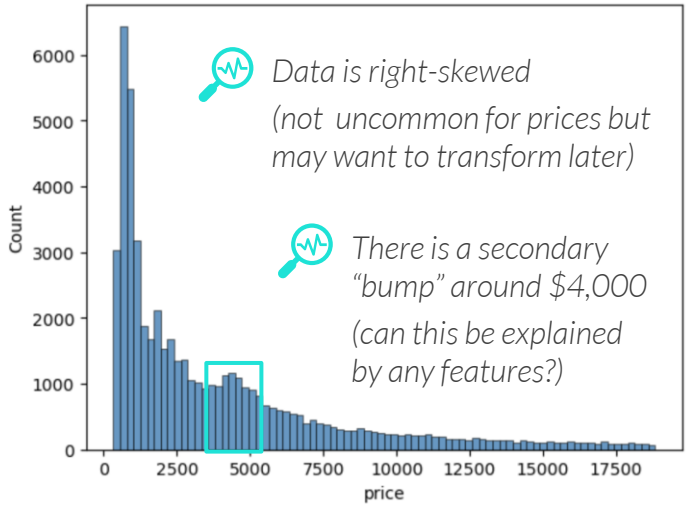
EXPLORING THE TARGET

Exploring the target variable lets you understand where most values lie, how spread out they are, and what shape, or distribution, they have

- Charts like **histograms** and **boxplots** are great tools to explore regression targets

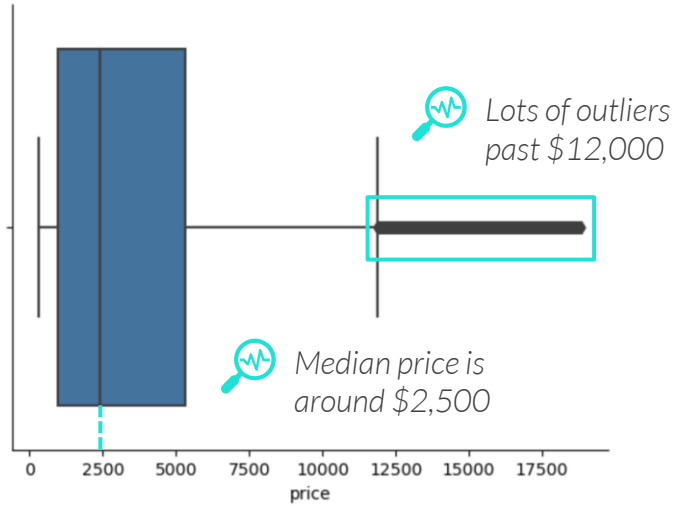
*The **seaborn** library is ideal for charts*

```
import seaborn as sns

sns.histplot(diamonds["price"]);
```




```
sns.boxplot(x=diamonds["price"])

sns.despine();
```

***sns.despine()** removes the top & right borders*



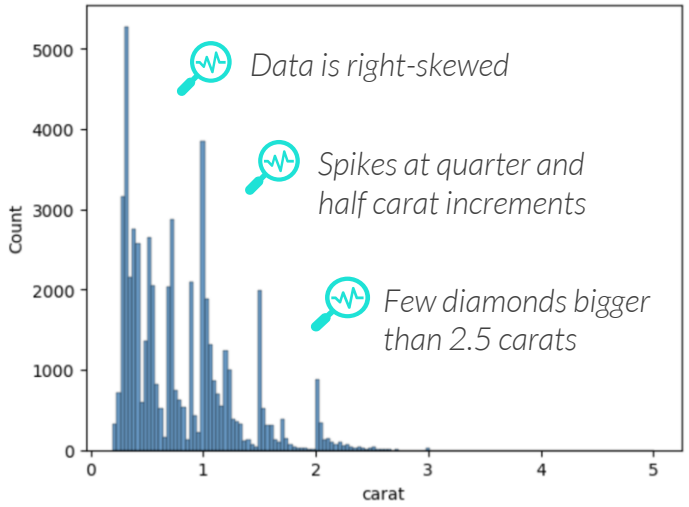
EXPLORING THE FEATURES

- EDA for Regression
- Exploring the Target
- Exploring the Features
- Linear Relationships
- Exploring Relationships
- Preparing for Modeling

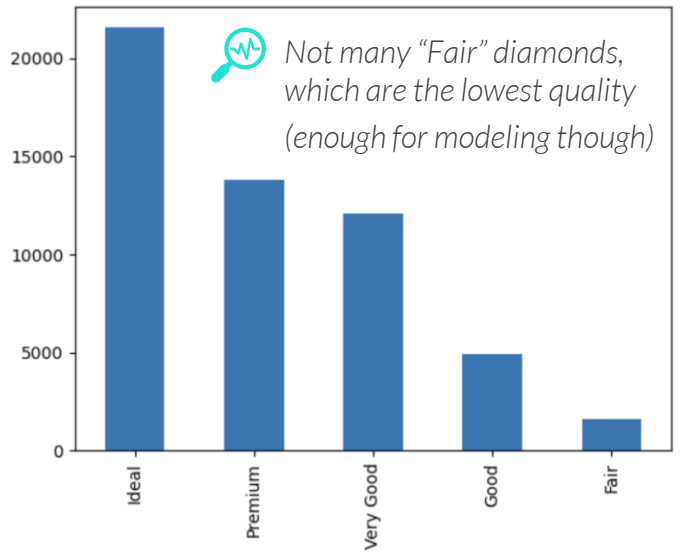
Exploring the features helps you understand them and start to get a sense of the transformations you may need to apply to each one

- **Histograms** and **boxplots** let you explore *numeric* features
- The **.value_counts()** method and **bar charts** are best for *categorical* features

```
sns.histplot(diamonds["carat"]);
```




```
diamonds["cut"].value_counts().plot.bar();
```



LINEAR RELATIONSHIPS

It's common for numeric variables to have **linear relationships** between them

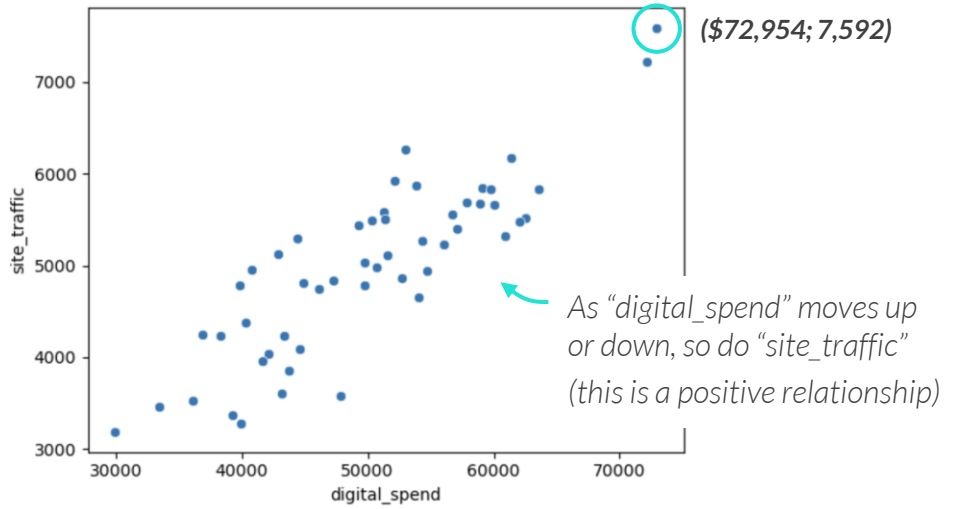
- When one variable changes, so does the other
- This relationship is commonly visualized with a **scatterplot**

EXAMPLE *Relationship between digital advertising spend and site traffic*

```
ads.head()
```

	week	offline_spend	digital_spend	site_traffic	site_load_time
0	1	35263	46125	4751	3.380876
1	2	53261	60007	5661	2.804510
2	3	43149	50314	5491	3.986187
3	4	47527	44432	5293	3.653095
4	5	37959	72954	7592	4.133515

```
sns.scatterplot(data=ads, x="digital_spend", y="site_traffic");
```

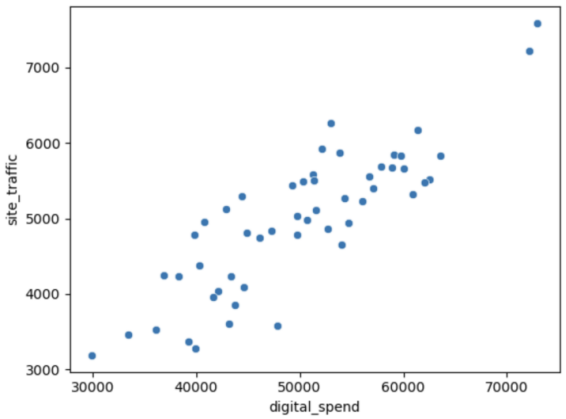


LINEAR RELATIONSHIPS

There are two possible linear relationships: **positive** & **negative**

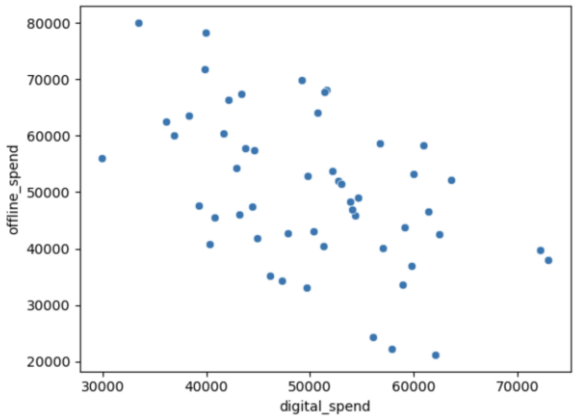
- Variables can also have **no relationship**

Positive Relationship



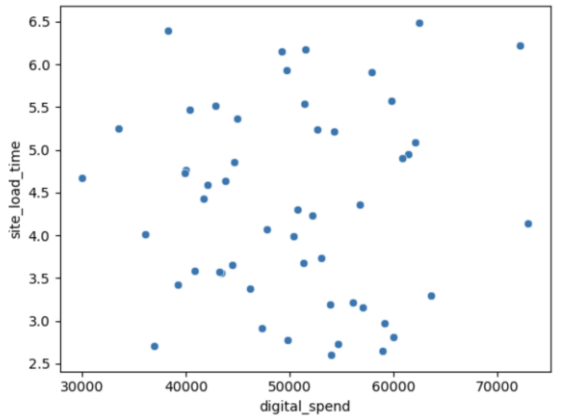
*As one changes, the other changes in the **same direction***

Negative Relationship



*As one changes, the other changes in the **opposite direction***

No Relationship



No association can be found between the changes in one variable and the other

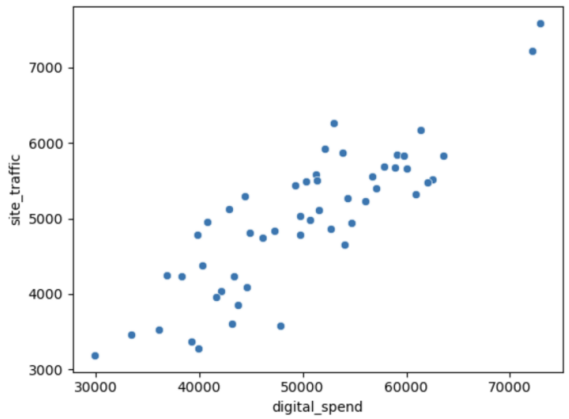
CORRELATION

- EDA for Regression
- Exploring the Target
- Exploring the Features
- Linear Relationships
- Exploring Relationships
- Preparing for Modeling

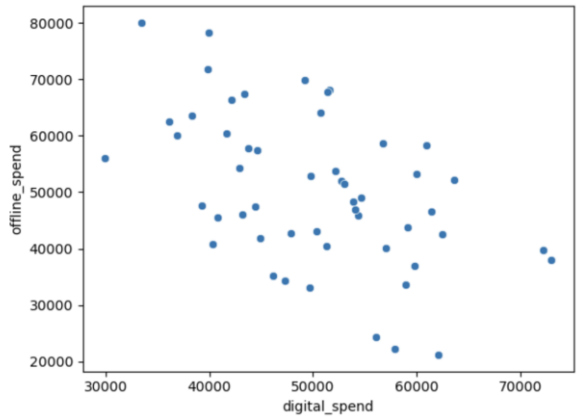
The **correlation (r)** measures the strength & direction of a linear relationship *(-1 to 1)*

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

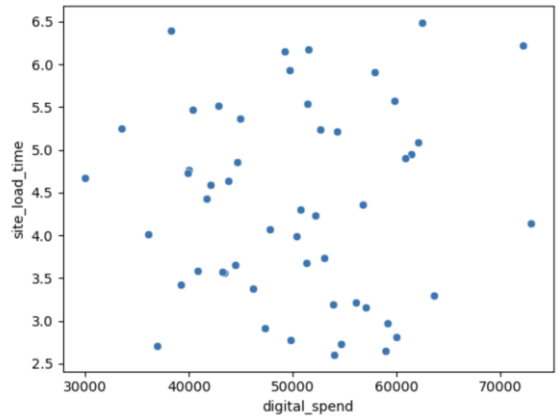
r = 0.858



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	

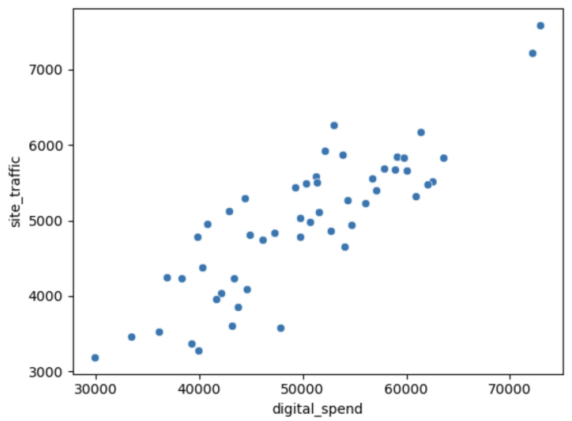
Negative Positive

CORRELATION

The **correlation (r)** measures the strength & direction of a linear relationship *(-1 to 1)*

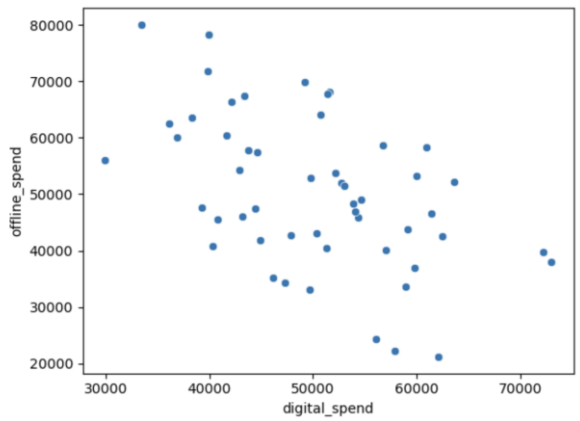
- You can use the **.corr()** method to calculate correlations in Pandas – df["col1"].corr(df["col2"])

r = 0.858



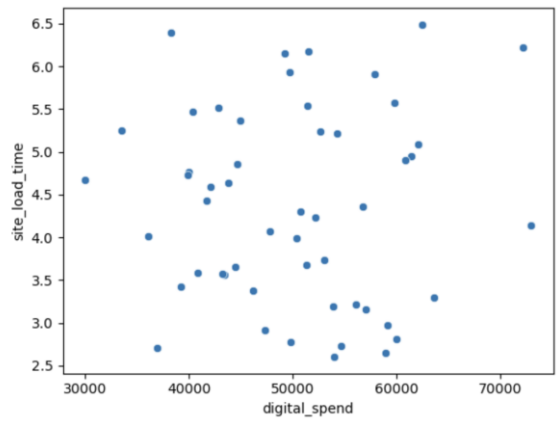
Strong positive correlation

r = -0.499



Moderate negative correlation

r = 0.008



No correlation

PRO TIP: Highly correlated variables tend to be the best candidates for your "baseline" regression model, but other variables can still be useful features after exploring non-linear relationships and fixing data issues

PRO TIP: CORRELATION MATRIX

- EDA for Regression
- Exploring the Target
- Exploring the Features
- Linear Relationships
- Exploring Relationships
- Preparing for Modeling

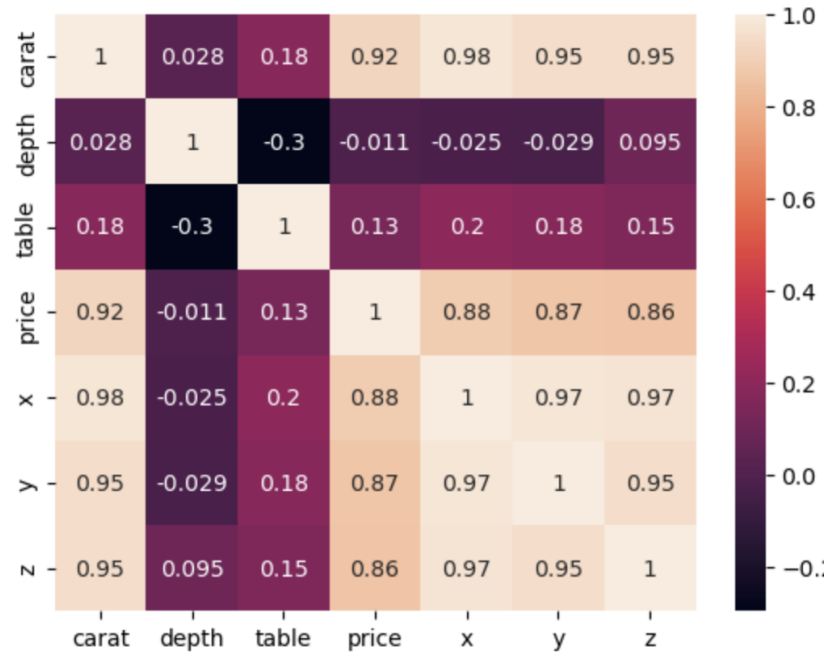
A **correlation matrix** returns the correlation between each column in a DataFrame

- Use **df.corr(numeric_only=True)** to only consider numeric columns in the DataFrame
- Wrap you correlation matrix in **sns.heatmap()** to make it easier to interpret

```
diamonds.corr(numeric_only=True)
```

	carat	depth	table	price	x	y	z
carat	1.000000	0.028234	0.181602	0.921591	0.975093	0.951721	0.953387
depth	0.028234	1.000000	-0.295798	-0.010630	-0.025289	-0.029340	0.094927
table	0.181602	-0.295798	1.000000	0.127118	0.195333	0.183750	0.150915
price	0.921591	-0.010630	0.127118	1.000000	0.884433	0.865419	0.861249
x	0.975093	-0.025289	0.195333	0.884433	1.000000	0.974701	0.970771
y	0.951721	-0.029340	0.183750	0.865419	0.974701	1.000000	0.952005
z	0.953387	0.094927	0.150915	0.861249	0.970771	0.952005	1.000000

```
sns.heatmap(diamonds.corr(numeric_only=True), annot=True);
```

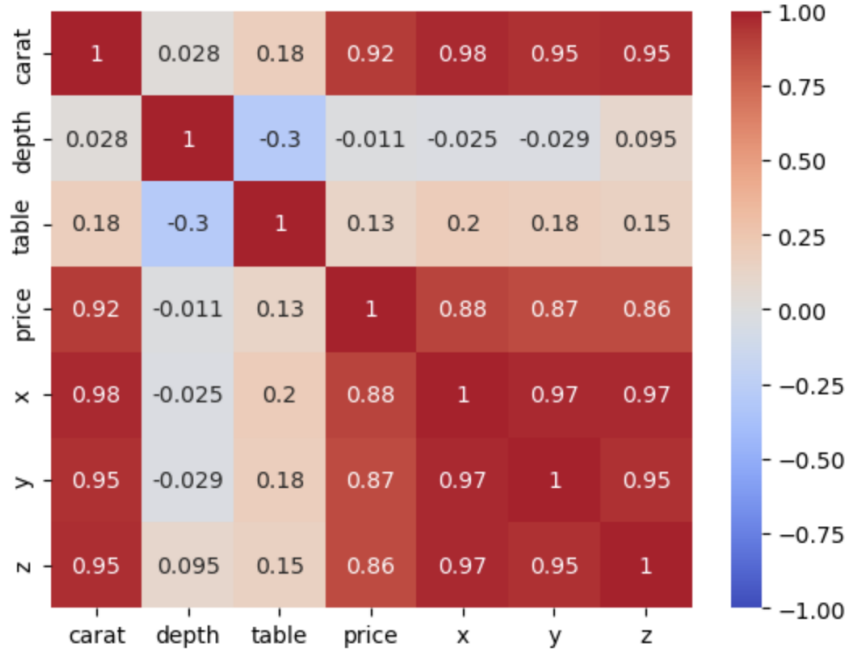


PRO TIP: CORRELATION MATRIX

A **correlation matrix** returns the correlation between each column in a DataFrame

- Use **df.corr(numeric_only=True)** to only consider numeric columns in the DataFrame
- Wrap you correlation matrix in **sns.heatmap()** to make it easier to interpret

```
sns.heatmap(diamonds.corr(numeric_only=True), annot=True, vmin=-1, vmax=1, cmap="coolwarm");
```



Setting the maximum and minimum values at -1 and 1 respectively, and adding a divergent color scale can help increase interpretability

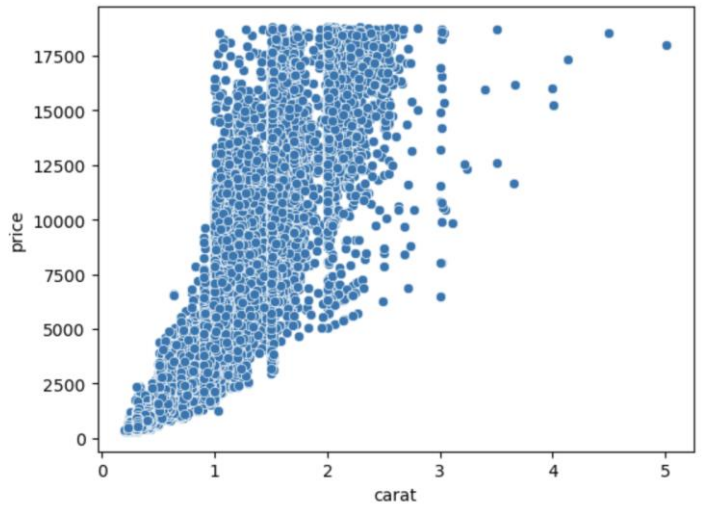
FEATURE-TARGET RELATIONSHIPS

- EDA for Regression
- Exploring the Target
- Exploring the Features
- Linear Relationships
- Exploring Relationships
- Preparing for Modeling

Exploring **feature-target relationships** helps identify potentially useful predictors

- Use scatterplots for numeric features and bar charts for categorical features

```
sns.scatterplot(diamonds, x="carat", y="price");
```

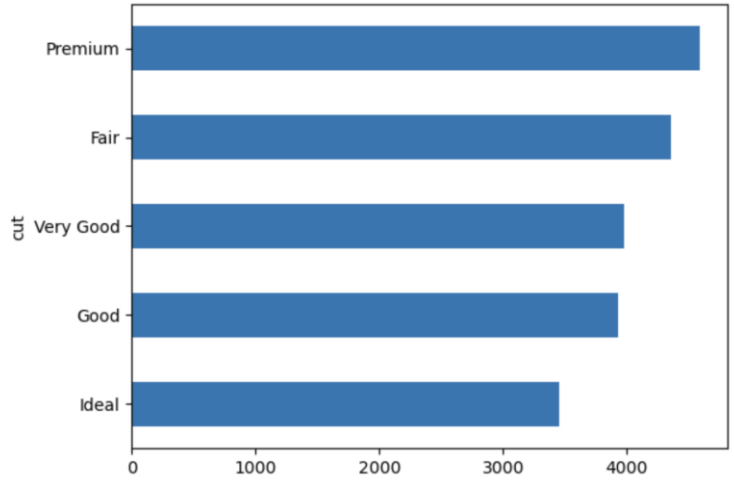



```
diamonds["carat"].corr(diamonds["price"])
```

```
0.9215912778016124
```

Strong positive relationship between carat & price (potentially exponential and not linear)

```
diamonds.groupby("cut")["price"].mean().sort_values().plot.barh();
```



Average prices differ between cuts:

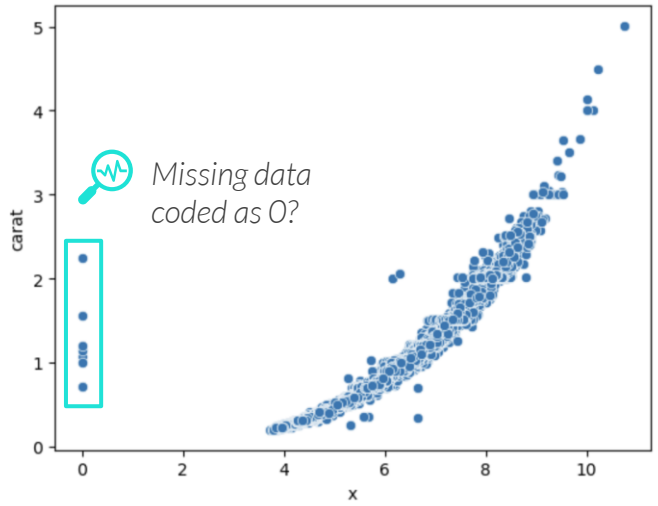
- *"Fair" cut diamonds (worst) have the 2nd highest average price*
- *"Ideal" cut diamonds (best) have the lowest average price*

FEATURE-FEATURE RELATIONSHIPS

- EDA for Regression
- Exploring the Target
- Exploring the Features
- Linear Relationships
- Exploring Relationships
- Preparing for Modeling

Exploring **feature-feature relationships** helps identify highly correlated features, which can cause problems with regression models *(more on this later!)*

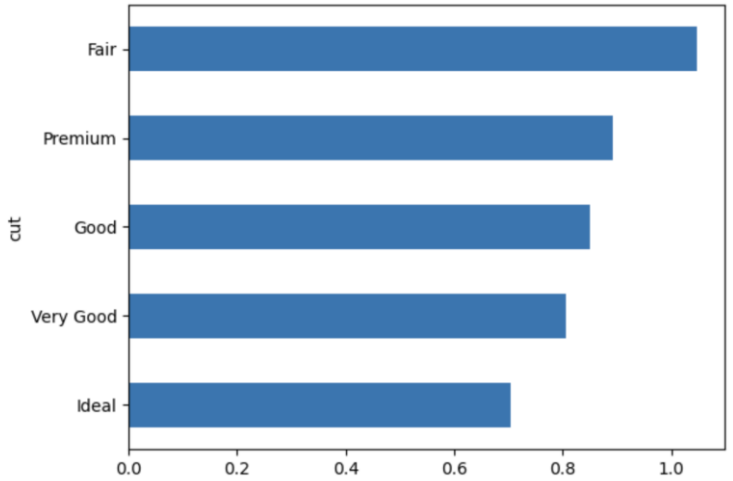
```
sns.scatterplot(diamonds, x="x", y="carat");
```



Strong positive relationship between diamond length (x) and carat

(we might not be able to include both features in our model – more later!)

```
diamonds.groupby("cut")["carat"].mean().sort_values().plot.barh();
```



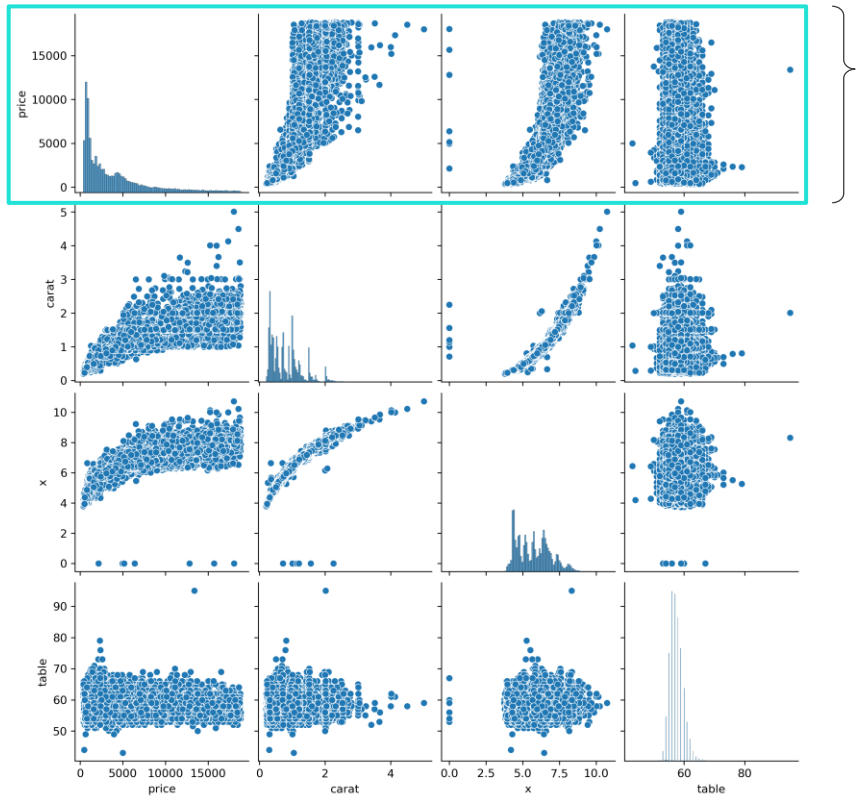
"Fair" cut diamonds have the largest average carat size, which explains why their average price is higher than "Ideal" cut diamonds, which have the smallest average carat size

PRO TIP: PAIRPLOTS

- EDA for Regression
- Exploring the Target
- Exploring the Features
- Linear Relationships
- Exploring Relationships
- Preparing for Modeling

Use **sns.pairplot()** to create a pairplot that shows all the scatterplots and histograms that can be made using the numeric variables in a DataFrame

```
sns.pairplot(diamonds);
```



*The row with your **target** can help explore numeric feature-target relationships quickly!*

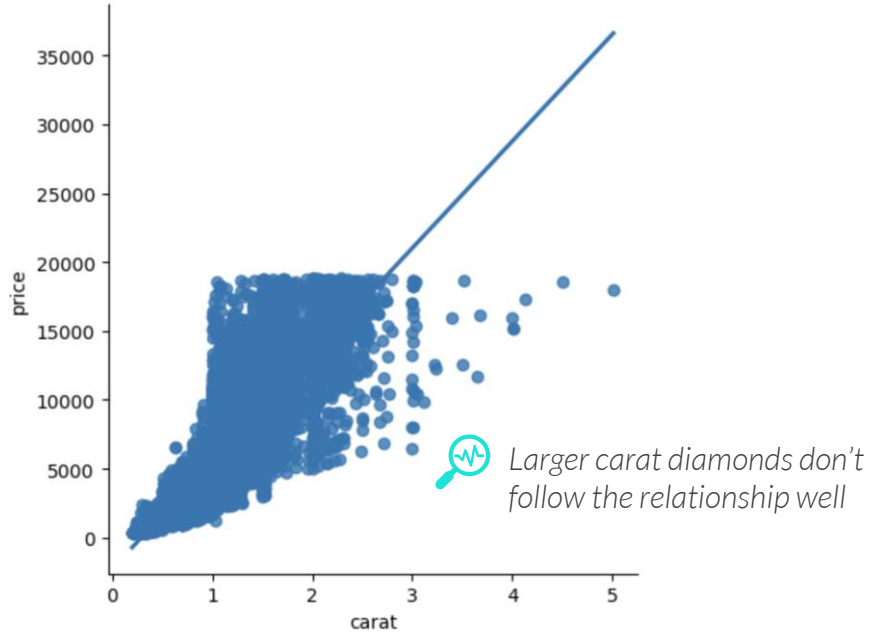
PRO TIP: It can take a long time to generate pairplots for large datasets, so consider using **df.sample()** to speed up the process significantly without losing high-level insights!

PRO TIP: LMPLOTS

Use **sns.lmplot()** to create a scatterplot with a fitted regression line *(more soon!)*

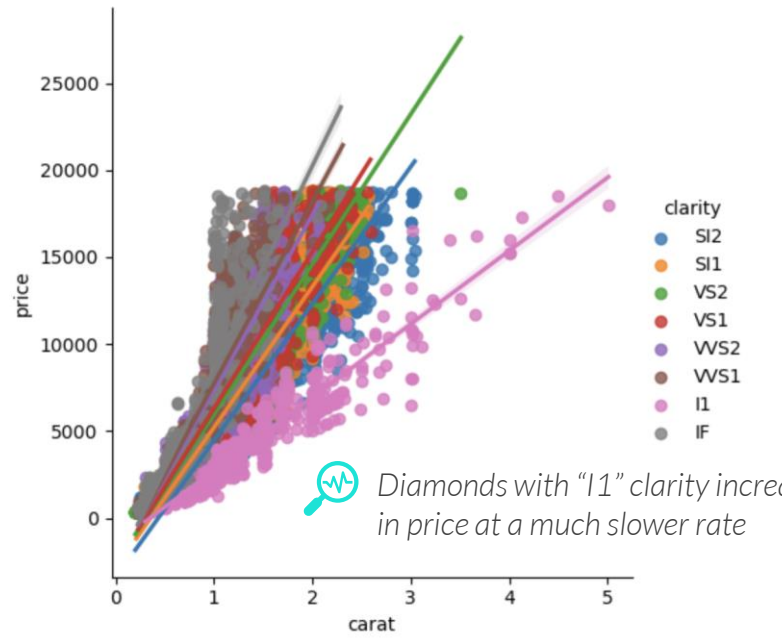
- This is commonly used to explore the impact of other variables on a linear relationship
- **sns.lmplot(df, x="feature", y="target", hue="categorical feature")**

```
sns.lmplot(diamonds, x="carat", y="price");
```



Larger carat diamonds don't follow the relationship well

```
sns.lmplot(diamonds, x="carat", y="price", hue="clarity");
```



Diamonds with "I1" clarity increase in price at a much slower rate

PREPARING FOR MODELING

- EDA for Regression
- Exploring the Target
- Exploring the Features
- Linear Relationships
- Exploring Relationships
- Preparing for Modeling

Preparing for modeling involves structuring the data as a valid input for a model:

- Stored in a single table *(DataFrame)*
- Aggregated to the right grain *(1 row per target)*
- Non-null *(no missing values)* and numeric

Data ready for EDA

Carat	Cut	Color	Price
0.90	Good	D	$3,423
0.31	Fair	E	$527
0.52	Very Good		$1,250
1.55	Ideal	D	$12,500

Data ready for modeling

Index	Carat	Cut	D	Missing	Price ($)
0	0.90	1	1	0	3423
1	0.31	0	0	0	527
2	0.52	2	0	1	1250
3	1.55	3	1	0	12500

Features *Target*

We'll revisit this during the **feature engineering** section of the course

KEY TAKEAWAYS

It's critical to **explore the features & target** in your data

- *Use histograms and boxplots to visualize and explore your target and any numeric features*
- *Use bar charts to visualize and explore categorical features*

Use scatterplots & correlations to **find linear relationships** in your data

- *Features that are highly correlated with your target are likely strong predictors of it*
- *Features that are highly correlated with each other can cause problems in a model*

Remember to **prepare you data for modeling** after performing EDA

- *The data must be stored in a single table, each column must be numeric, and missing values must be handled*

SIMPLE LINEAR REGRESSION

SIMPLE LINEAR REGRESSION

In this section we'll build our first **simple linear regression** models in Python and learn about the metrics and statistical tests that help evaluate their quality and output

TOPICS WE'LL COVER:

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

GOALS FOR THIS SECTION:

- Introduce the linear regression equation
- Visualize the the least squared error method for finding the line of best fit
- Build linear regression models in Python and use them to make predictions for the target
- Interpret the model summary statistics and use them to evaluate the model's accuracy

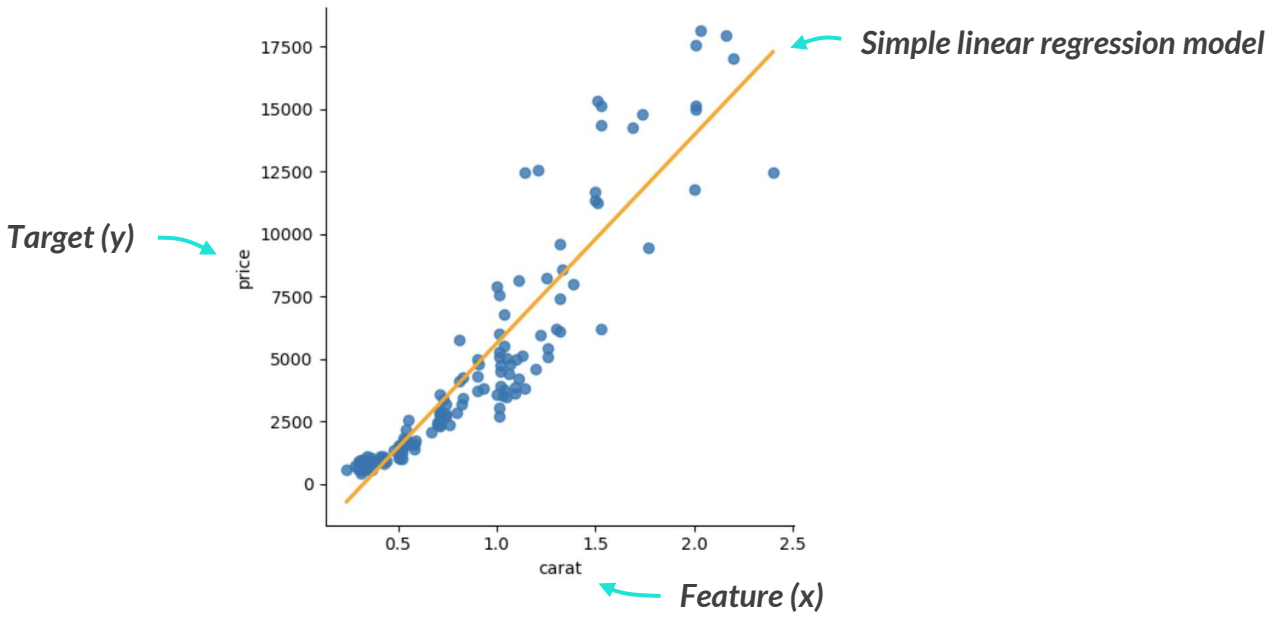
SIMPLE LINEAR REGRESSION

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

Simple linear regression models use a *single feature* to predict the target

- This is achieved by fitting a line through the data points in a scatterplot *(like an lmplot!)*

EXAMPLE | *Simple linear regression model using carat and price*



LINEAR REGRESSION MODEL

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

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

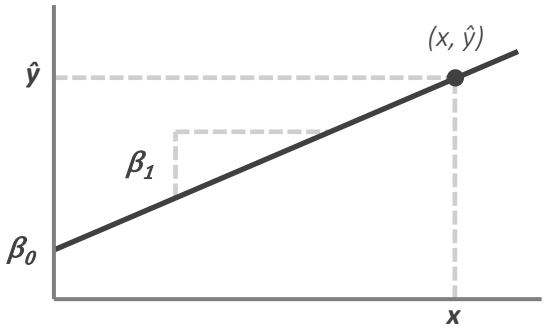
*The **predicted** value for the target*

The value for the feature

$$\hat{y} = \beta_0 + \beta_1 x$$

*The **y-intercept***

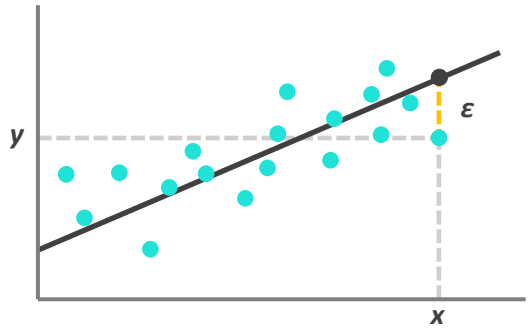
*The **slope** of the relationship*

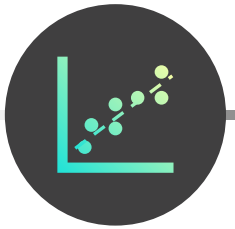


*The **actual** value for the target*

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

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





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!)*

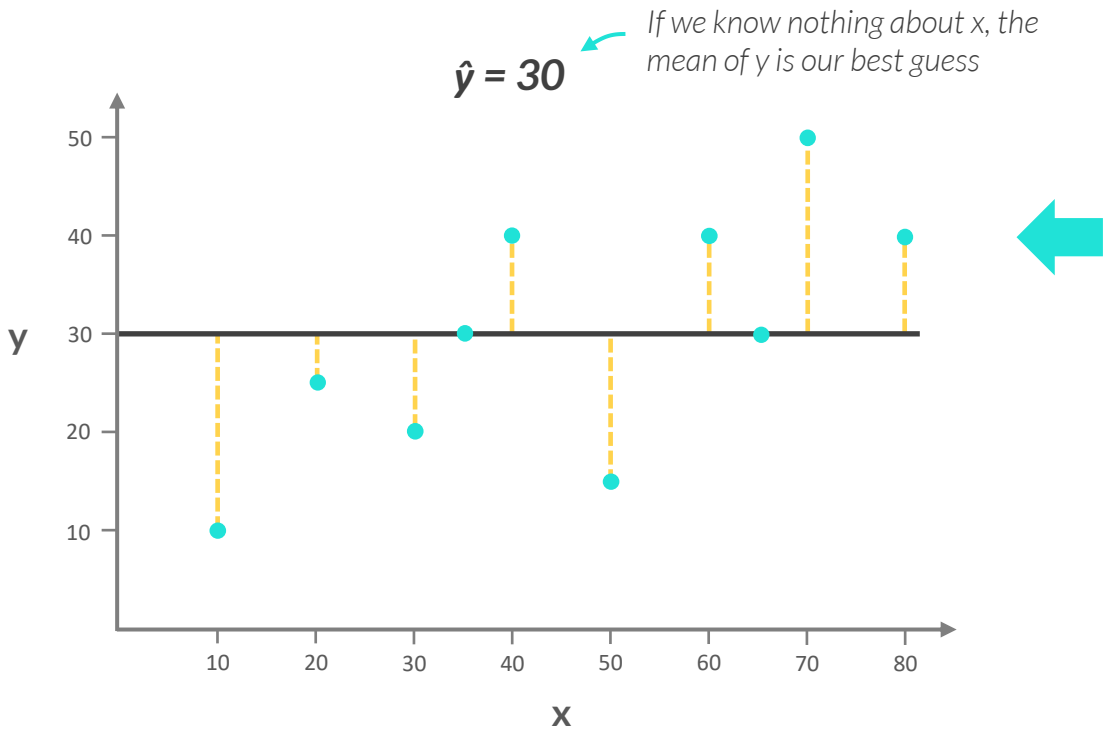
Ordinary Least Squares (OLS) is another term for traditional linear regression

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**



x	y
10	10
20	25
30	20
35	30
40	40
50	15
60	40
65	30
70	50
80	40

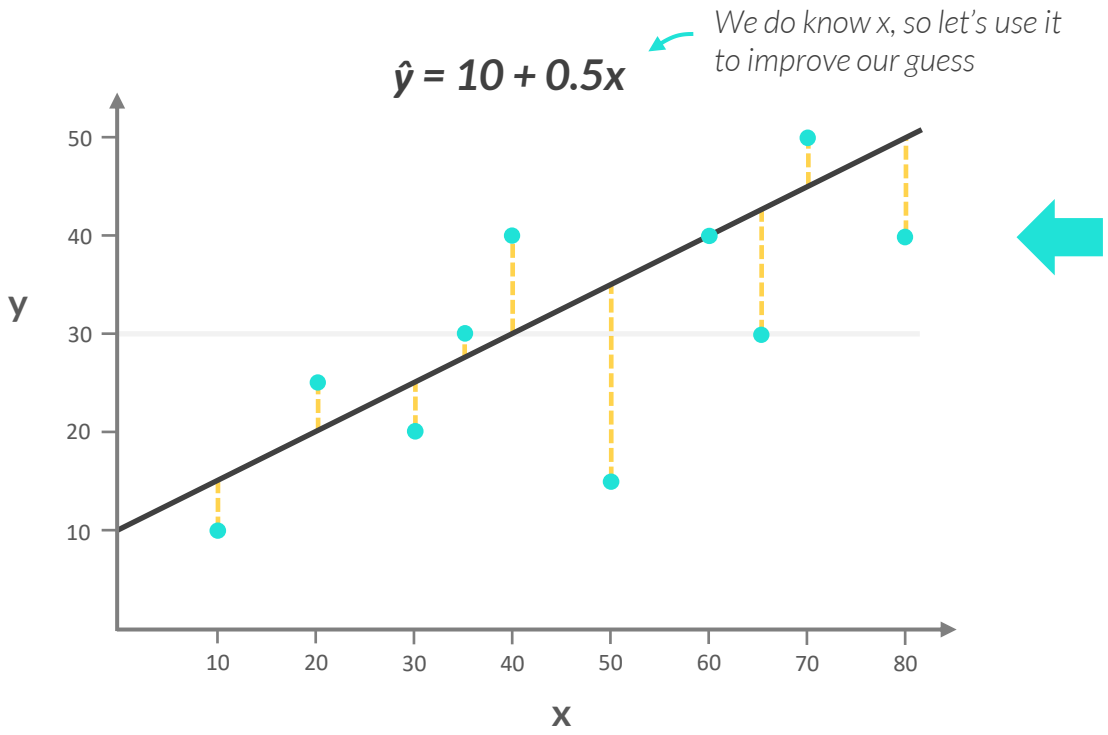
SUM OF SQUARED ERROR: = **1,450**

LEAST SQUARED ERROR

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

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**



x	y	ŷ	ε	ε^2
10	10	15	5	**25**
20	25	20	-5	**25**
30	20	25	5	**25**
35	30	27.5	-2.5	**6.25**
40	40	30	-10	**100**
50	15	35	20	**400**
60	40	40	0	**0**
65	30	42.5	12.5	**156.25**
70	50	45	-5	**25**
80	40	50	10	**100**
				II

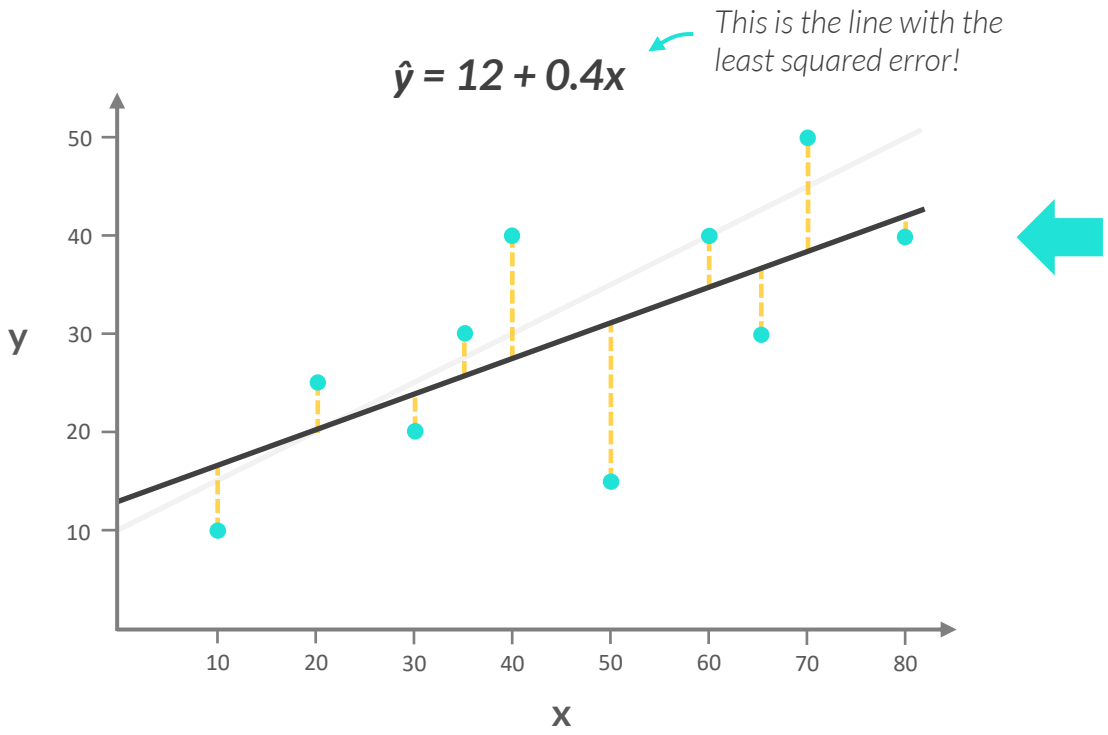
SUM OF SQUARED ERROR: **862.5**

LEAST SQUARED ERROR

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

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**



x	y	ŷ	ε	ε^2
10	10	16	6	**36**
20	25	20	-5	**25**
30	20	24	4	**16**
35	30	26	-4	**16**
40	40	28	-12	**144**
50	15	32	17	**289**
60	40	36	-4	**16**
65	30	38	8	**64**
70	50	40	-10	**100**
80	40	44	4	**16**
				II

SUM OF SQUARED ERROR: **722**

REGRESSION IN PYTHON

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

statsmodels

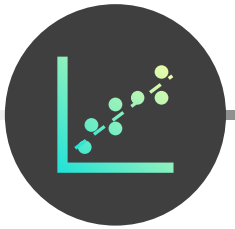
- **Ideal if your goal is inference**
- Similar output to other tools (SAS, R, Excel)
- Easy access to dozens of statistical tests
- Harder to leverage in production ML

scikit learn
machine learning in Python

- **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 Predictions
- Evaluation Metrics

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

```
import statsmodels.api as sm

X = sm.add_constant(diamonds["carat"])
y = diamonds["price"]

model = sm.OLS(y, X).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 Predictions
- Evaluation Metrics

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

```
import statsmodels.api as sm

X = sm.add_constant(diamonds["carat"])
y = diamonds["price"]

model = sm.OLS(y, X).fit()

model.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.849
Model:	OLS	**Adj. R-squared:**	0.849
Method:	Least Squares	**F-statistic:**	3.041e+05
Date:	Wed, 02 Aug 2023	**Prob (F-statistic):**	0.00
Time:	10:47:03	**Log-Likelihood:**	-4.7276e+05
No. Observations:	53943	**AIC:**	9.455e+05
Df Residuals:	53941	**BIC:**	9.455e+05
Df Model:	1		
Covariance Type:	nonrobust		

Model summary statistics

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2256.3950	13.055	-172.840	0.000	-2281.983	-2230.807
carat	7756.4362	14.066	551.423	0.000	7728.866	7784.006

Variable summary statistics

Omnibus:	14027.005	**Durbin-Watson:**	1.999
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	153060.389
Skew:	0.939	**Prob(JB):**	0.00
Kurtosis:	11.036	**Cond. No.**	3.65

Residual (error) statistics

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

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 Predictions
- Evaluation Metrics

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

```
import statsmodels.api as sm

X = sm.add_constant(diamonds["carat"])
y = diamonds["price"]

model = sm.OLS(y, X).fit()

model.summary()
```

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2256.3950	13.055	-172.840	0.000	-2281.983	-2230.807
carat	7756.4362	14.066	551.423	0.000	7728.866	7784.006

$$\hat{y} = -2256 + 7756x$$

How do we interpret this?

- An increase of 1 carat in a diamond is *associated* with a $7,756 dollar increase in its price
- We **cannot** say 1 carat *causes* a $7,756 increase in price without a more rigorous experiment
- Technically, a 0-carat diamond is predicted to cost -$2,256 dollars

MAKING PREDICTIONS

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

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

```
import statsmodels.api as sm

X = sm.add_constant(diamonds["carat"])
y = diamonds["price"]

model = sm.OLS(y, X).fit()

model.summary()
```

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2256.3950	13.055	-172.840	0.000	-2281.983	-2230.807
carat	7756.4362	14.066	551.423	0.000	7728.866	7784.006

```
model.predict([1, 1.5])
```

```
array([9378.25919172])
```

The 1 adds a constant

$$-2256 + 7756(\mathbf{1.5}) = \mathbf{9378}$$

A 1.5 carat diamond has a predicted price of $9,378

MAKING PREDICTIONS

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

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

```
import statsmodels.api as sm

X = sm.add_constant(diamonds["carat"])
y = diamonds["price"]

model = sm.OLS(y, X).fit()

model.summary()
```

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2256.3950	13.055	-172.840	0.000	-2281.983	-2230.807
carat	7756.4362	14.066	551.423	0.000	7728.866	7784.006

```
carats = sm.add_constant(pd.DataFrame({"carat": [.5, 1, 1.5, 2, 2.5]}))
model.predict(carats)
```

```
0     1621.823032
1     5500.041112
2     9378.259192
3    13256.477271
4    17134.695351
dtype: float64
```

R-SQUARED

- Linear Regression Model
- Least Squared Error
- Regression in Python
- Making Predictions
- 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 = sm.add_constant(diamonds["carat"])
y = diamonds["price"]

model = sm.OLS(y, X).fit()

model.summary()
```

```
diamonds["carat"].corr(diamonds["price"])**2
```

```
0.8493304833200086
```

Squaring the correlation *between the feature and target yields R^2 in simple linear regression (this doesn't hold in multiple linear regression)*

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.849
Model:	OLS	**Adj. R-squared:**	0.849
Method:	Least Squares	**F-statistic:**	3.041e+05
Date:	Wed, 02 Aug 2023	**Prob (F-statistic):**	0.00
Time:	10:47:03	**Log-Likelihood:**	-4.7276e+05
No. Observations:	53943	**AIC:**	9.455e+05
Df Residuals:	53941	**BIC:**	9.455e+05
Df Model:	1		
Covariance Type:	nonrobust		

*The model explains **84.9%** of the variation in price not explained by the mean of price*

R-SQUARED

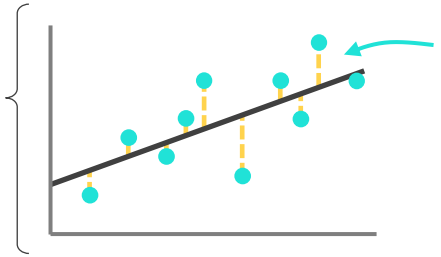
- Linear Regression Model
- Least Squared Error
- Regression in Python
- Making Predictions
- Evaluation Metrics

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

$$R^2 = 1 - \frac{SSE}{SST}$$

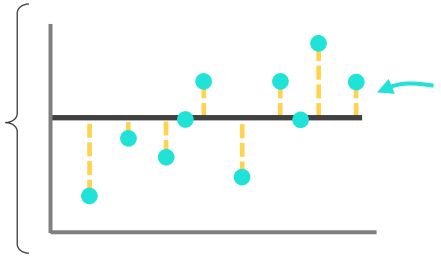
*This is the **sum of squared error** (distance between values & line)*



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

- *Reducing error will increase R^2*
- *A perfect model has an error of 0, or R^2 of 1*

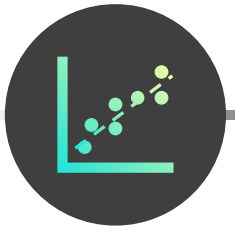
*This is the **sum of squared total** (distance between values & mean)*



This is the variation of "y" around its mean

- *If R^2 = 0, the model is no better than the mean of y*

A "good" value for R^2 is relative to your data – an R^2 of .05 might be industry leading in sports analytics, but if you were trying to prove a physics theory with an experiment, an R^2 of .95 might be a disappointment!



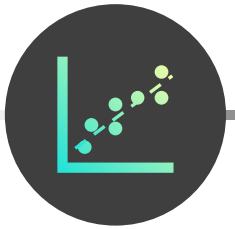
HYPOTHESIS TEST

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 **p-value**
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:

- **H_o: F=0** – the model is NOT significantly better than the mean *(naïve guess)*
- **H_a: F≠0** – the model is significantly better than the mean *(naïve 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

- This is set by **alpha (α)**, 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

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 = sm.add_constant(diamonds["carat"])
y = diamonds["price"]

model = sm.OLS(y, x).fit()

model.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.849
Model:	OLS	**Adj. R-squared:**	0.849
Method:	Least Squares	**F-statistic:**	3.041e+05
Date:	Wed, 02 Aug 2023	**Prob (F-statistic):**	0.00
Time:	10:47:03	**Log-Likelihood:**	-4.7276e+05
No. Observations:	53943	**AIC:**	9.455e+05
Df Residuals:	53941	**BIC:**	9.455e+05
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 Predictions
- Evaluation Metrics

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

- **$p \leq \alpha$** – reject the null hypothesis *(you're confident the model isn't useless)*
- **$p > \alpha$** – don't reject it *(the model is probably useless, and needs more training)*

```
import statsmodels.api as sm

x = sm.add_constant(diamonds["carat"])
y = diamonds["price"]

model = sm.OLS(y, x).fit()

model.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.849
Model:	OLS	**Adj. R-squared:**	0.849
Method:	Least Squares	**F-statistic:**	3.041e+05
Date:	Wed, 02 Aug 2023	**Prob (F-statistic):**	0.00
Time:	10:47:03	**Log-Likelihood:**	-4.7276e+05
No. Observations:	53943	**AIC:**	9.455e+05
Df Residuals:	53941	**BIC:**	9.455e+05
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 (carat is a good predictor of a diamond's price!)

T-STATISTICS & P-VALUES

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 = sm.add_constant(diamonds["carat"])
y = diamonds["price"]

model = sm.OLS(y, x).fit()

model.summary()
```

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2256.3950	13.055	-172.840	0.000	-2281.983	-2230.807
carat	7756.4362	14.066	551.423	0.000	7728.866	7784.006

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

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

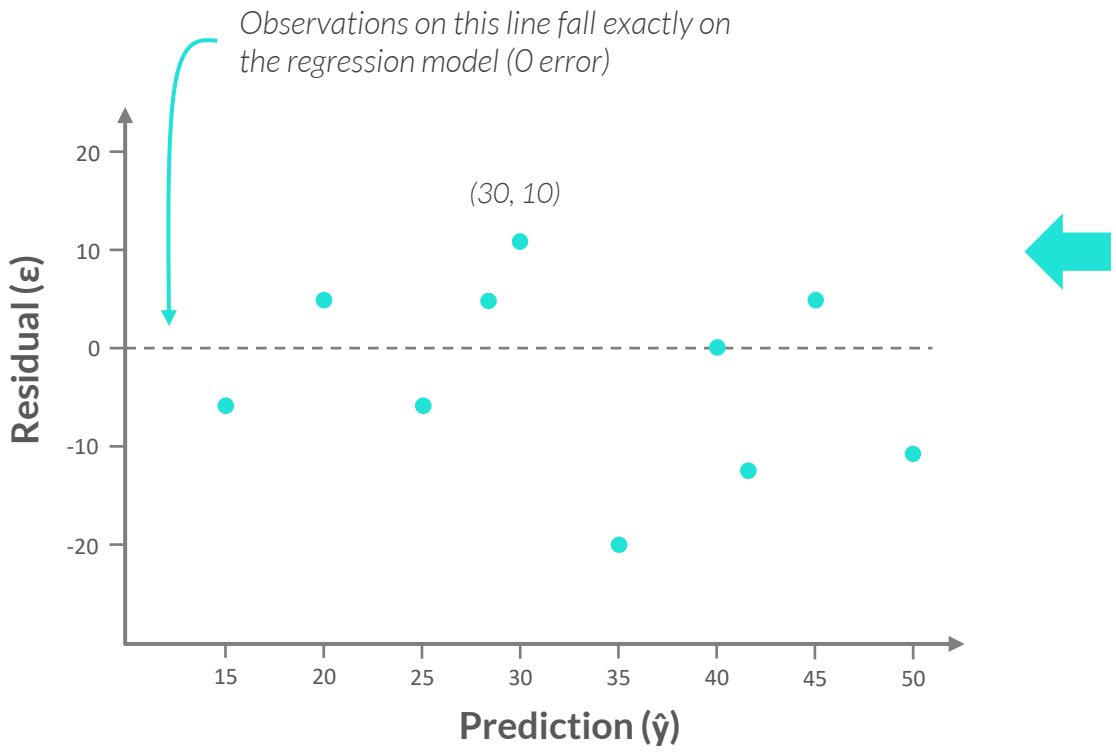
This will become more relevant when performing **variable selection** in multiple linear regression models (up next!)

RESIDUAL PLOTS

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

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

- Ideally, residual plots should be normally distributed around 0



x	y	ŷ	ε	ε^2
10	10	**15**	**-5**	25
20	25	**20**	**5**	25
30	20	**25**	**-5**	25
35	30	**27.5**	**2.5**	6.25
40	40	**30**	**10**	100
50	15	**35**	**-20**	400
60	40	**40**	**0**	0
65	30	**42.5**	**-12.5**	156.25
70	50	**45**	**5**	25
80	40	**50**	**-10**	100

RESIDUAL PLOTS

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

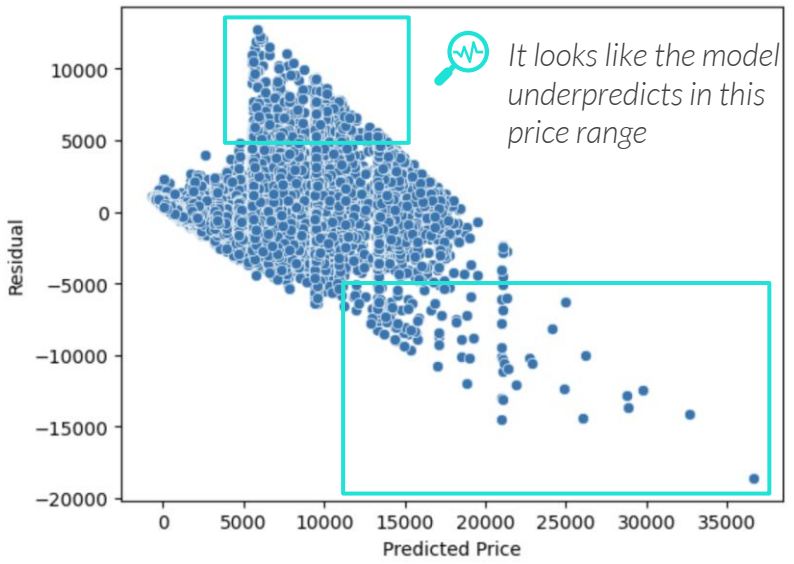
Residual plots show how well a model performs across the range of predictions

- 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");
```



CASE STUDY: HEALTH INSURANCE

THE SITUATION

You've just been hired as a Data Science Intern for **Maven National Insurance**, a large private health insurance provider in the United States

THE ASSIGNMENT

The company is looking at updating their **insurance pricing model** and want you to start a new one from scratch using only a handful of variables

If you're successful, they can reduce the complexity of their model while maintaining its accuracy *(the data you've been provided has already been QA'd and cleaned)*

THE OBJECTIVES

1. Identify the strongest predictor of insurance prices using correlation
2. Build a simple linear regression model using this feature
3. Predict insurance prices for new customers using the model

ASSIGNMENT: SIMPLE LINEAR REGRESSION

NEW MESSAGE

June 30, 2023

From: **Cam Correlation** *(Sr. Data Scientist)*

Subject: **Price predictions**

Hi there!

Looks a few variables are promising!

Can you build a linear regression model with the target as "price" and the most correlated variable as the feature?

I'd also like to see a few predictions for common values of the feature that you selected.

Thanks!

02_simple_regression_assignments.ipynb

Reply Forward

Key Objectives

1. Build a simple linear regression model with "price" as the target and the most correlated variable as the feature
2. Interpret the model summary
3. Visualize the model residuals
4. Predict new "price" values with the model

KEY TAKEAWAYS

A **simple linear regression** model is the line that best fits a scatterplot

- *The line can be described using an equation with a slope and y-intercept, plus an error term*
- *The least squared error method is used to find the line of best fit*

Python uses the **statsmodels** & **scikit-learn** libraries to fit regression models

- *Statsmodels is ideas if your goal is inference, while scikit-learn is optimal for prediction workflows*

Linear regression models can be used to **predict new data**

- *These predictions can be used to assess if our model performance holds on new data and help make decisions*

The model summary contains metrics used to **evaluate the model**

- *The model's R^2 value and the F-test's p-value help determine if the regression model is useful*

MULTIPLE LINEAR REGRESSION

MULTIPLE LINEAR REGRESSION

In this section we'll build **multiple linear regression** models using more than one feature, evaluate the model fit, perform variable selection, and compare models using error metrics

TOPICS WE'LL COVER:

- Multiple Regression
- Variable Selection
- Mean Error Metrics

GOALS FOR THIS SECTION:

- Learn how to fit and interpret multiple linear regression models in Python
- Walk through methods of performing variable selection, ensuring model variables add value
- Compare the predictive accuracy across models using mean error metrics like MAE and RMSE

MULTIPLE LINEAR REGRESSION MODEL

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:

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

MULTIPLE LINEAR REGRESSION MODEL:

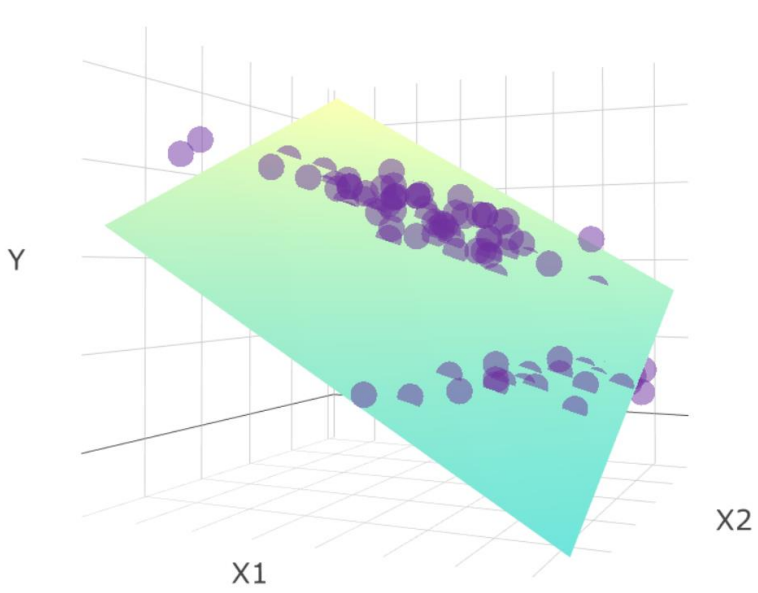
$$y = \beta_0 + \underbrace{\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_k x_k}_{} + \epsilon$$

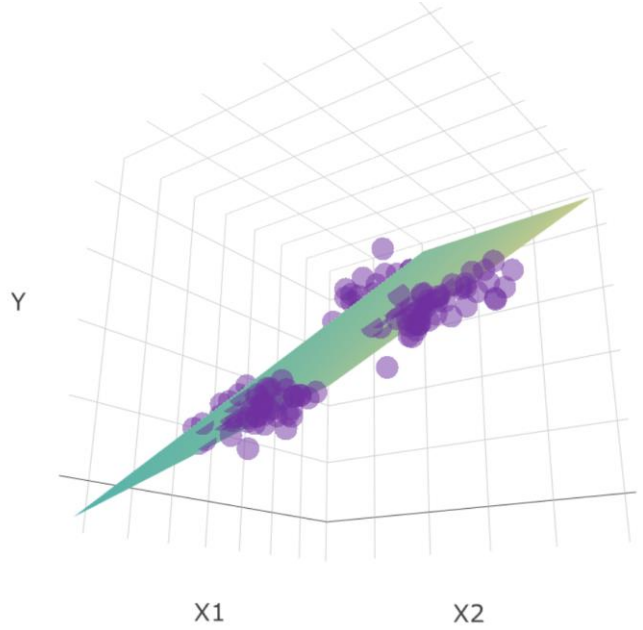
*Instead of just one "x", we have a **whole set of features** (and associated coefficients) to help predict the target (y)*

MULTIPLE LINEAR REGRESSION MODEL

- Multiple Regression
- Variable Selection
- Mean Error Metrics

To visualize how a multiple linear regression model works with 2 features, imagine fitting a plane *(instead of a line)* through a 3D scatterplot:





Multiple regression can scale well beyond 2 variables, but this is where visual analysis breaks down – and one reason why we need **machine learning**!

FITTING A MULTIPLE REGRESSION

- Multiple Regression
- Variable Selection
- Mean Error Metrics

To **fit a multiple regression**, include the desired features in the "X" DataFrame

```
diamonds.corr(numeric_only=True)
```

	carat	depth	table	price	x	y	z
carat	1.000000	0.028234	0.181602	0.921591	0.975093	0.951721	0.953387
depth	0.028234	1.000000	-0.295798	-0.010630	-0.025289	-0.029340	0.094927
table	0.181602	-0.295798	1.000000	0.127118	0.195333	0.183750	0.150915
price	0.921591	-0.010630	0.127118	1.000000	0.884433	0.865419	0.861249
x	0.975093	-0.025289	0.195333	0.884433	1.000000	0.974701	0.970771
y	0.951721	-0.029340	0.183750	0.865419	0.974701	1.000000	0.952005
z	0.953387	0.094927	0.150915	0.861249	0.970771	0.952005	1.000000

"carat" and "x" are the most correlated features with "price"

```
X = sm.add_constant(diamonds[["carat", "x"]])
y = diamonds["price"]

model = sm.OLS(y, X).fit()

model.summary()
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.853
Model:	OLS	**Adj. R-squared:**	0.853
Method:	Least Squares	**F-statistic:**	1.570e+05
Date:	Thu, 03 Aug 2023	**Prob (F-statistic):**	0.00
Time:	10:05:44	**Log-Likelihood:**	-4.7201e+05
No. Observations:	53943	**AIC:**	9.440e+05
Df Residuals:	53940	**BIC:**	9.441e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1738.1187	103.618	16.774	0.000	1535.026	1941.211
carat	1.013e+04	62.551	161.886	0.000	1e+04	1.02e+04
x	-1026.9048	26.432	-38.851	0.000	-1078.711	-975.099

Omnibus:	14014.880	**Durbin-Watson:**	2.001
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	317574.692
Skew:	0.717	**Prob(JB):**	0.00
Kurtosis:	14.800	**Cond. No.**	112.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

R^2 increased from 0.849

$p < 0.05$, so the model isn't useless

$p < 0.05$, so all variables are useful

"x" has a negative coefficient but a positive correlation (this is a concern we'll cover shortly)

INTERPRETING MULTIPLE REGRESSION

Interpreting multiple regression is like simple regression, with a slight twist

- Multiple Regression
- Variable Selection
- Mean Error Metrics

```
X = sm.add_constant(diamonds[["carat", "x"]])
y = diamonds["price"]

model = sm.OLS(y, X).fit()

model.summary()
```

	coef	std err	t	P>|t|	[0.025	0.975]
const	1738.1187	103.618	16.774	0.000	1535.026	1941.211
carat	1.013e+04	62.551	161.886	0.000	1e+04	1.02e+04
x	-1026.9048	26.432	-38.851	0.000	-1078.711	-975.099

$$\hat{y} = 1738 + 10130(carat) - 1027(x)$$

How do we interpret this?

- Technically, a 0-carat diamond with 0 length (x) is predicted to cost $1,738 dollars
- An increase in carat by 1 corresponds to a $10,130 increase in price, **holding x constant**
- An increase in x by 1 corresponds to a $1,027 decrease in price, **holding carat constant**

VARIABLE SELECTION

Multiple Regression

Variable Selection

Mean Error Metrics

Variable selection is a critical part of the modeling process that helps reduce complexity by only including features that help meet the model's goal

- Do new variables improve model accuracy? *(critical for prediction)*
- Is each variable statistically significant? *(critical for inference)*
- Do new variables make the model more challenging to interpret or explain to stakeholders?

EXAMPLE | *Using all the possible features to predict diamond price*

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.859
Model:	OLS	**Adj. R-squared:**	0.859

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.085e+04	447.545	46.584	0.000	2e+04	2.17e+04
carat	1.069e+04	63.198	169.094	0.000	1.06e+04	1.08e+04
depth	-203.1414	5.504	-36.909	0.000	-213.929	-192.354
table	-102.4407	3.084	-33.217	0.000	-108.485	-96.396
x	-1315.7397	43.069	-30.550	0.000	-1400.155	-1231.324
y	66.3300	25.522	2.599	0.009	16.306	116.354
z	41.6285	44.303	0.940	0.347	-45.207	128.464

R^2 improved to .859 compared to the simple regression model (.849)
If the goal is inference, a single variable model may be a good choice!

"x" still has a negative coefficient
This helps R^2 but makes the model hard to explain

"z" has a p-value greater than alpha (0.347 > 0.05)
We should drop this variable from the model

ADJUSTED R-SQUARED

Multiple Regression

Variable Selection

Mean Error Metrics

A criticism of r-squared is that it will never decrease as new variables are added

Adjusted r-squared corrects this by penalizing new variables added to a model

- This measure has no meaning whatsoever, but it helps as a variable selection tool

EXAMPLE | *Adding a random column to a sample of 100 diamonds*

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.782
Model:	OLS	**Adj. R-squared:**	0.780
Method:	Least Squares	**F-statistic:**	2501

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2517.2645	355.136	-7.088	0.000	-3222.020	-1812.509
carat	8631.6433	459.978	18.765	0.000	7718.831	9544.455

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.784
Model:	OLS	**Adj. R-squared:**	0.780
Method:	Least Squares	**F-statistic:**	176.2

R^2 increased but adjusted R^2 didn't

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2263.3000	452.073	-5.006	0.000	-3160.540	-1366.060
carat	8673.9033	462.726	18.745	0.000	7755.520	9592.287
random	-5.5127	6.063	-0.909	0.366	-17.547	6.521

In this case, the p-value could also tell us to remove this variable (other times, a variable can be significant and lower adjusted R^2)

ASSIGNMENT: MULTIPLE LINEAR REGRESSION

NEW MESSAGE

July 2, 2023

From: **Cam Correlation** *(Sr. Data Scientist)*

Subject: **Multiple Regression Model**

Hi there!

Let's include more features in the model.

Can you start by adding all of the PC component variables, ram, screen, hd, and speed?

Then take a look at including trend, which accounts for cost decreases over time, and ads, which measure marketing spend on each model.

Thanks!

03_multiple_regression_assignments.ipynb

Reply Forward

Key Objectives

1. Build and interpret two multiple linear regression models
2. Evaluate model fit and coefficient values for both models

MEAN ERROR METRICS

Multiple Regression

Variable Selection

Mean Error Metrics

Mean error metrics measure how well your regression model *fits in in the units of our target*, as opposed to how well it explains variance (like R-Squared)

- The most common are **Mean Absolute Error** (MAE) and **Root Mean Squared Error** (RMSE)
- They are used to compare model fit across models *(the lower the better!)*

MAE

*Average of the **absolute** distance between actual & predicted values*

$$\frac{\sum_i |y_i - \hat{y}_i|}{n}$$

MSE

*Average of the **squared** distance between actual & predicted values*

$$\frac{\sum_i (y_i - \hat{y}_i)^2}{n}$$

RMSE

***Square root** of Mean Squared Error, to return to the target's units (like MAE)*

$$\sqrt{\frac{\sum_i (y_i - \hat{y}_i)^2}{n}}$$

RMSE is more sensitive to large outliers, so it is preferred over MAE in situations where they are particularly undesirable

MEAN ERROR METRICS

Multiple Regression

Variable Selection

Mean Error Metrics

You can use **sklearn.metrics** to calculate MAE and MSE for your model

- sklearn.metrics.**mean_absolute_error**(y_actual, y_predicted)
- sklearn.metrics.**mean_squared_error**(y_actual, y_predicted)

```
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import mean_squared_error as rmse

X = sm.add_constant(diamonds["carat"])
y = diamonds["price"]

model = sm.OLS(y, X).fit()

print(f"MAE: {mae(y, model.predict())}")
print(f"RMSE {rmse(y, model.predict(), squared=False)}")
```

```
MAE: 1007.4339350399421
RMSE 1548.4940983743945
```

*This returns **RMSE** instead of MSE*

The simple linear regression model has an average prediction error of ~$1,000

The outliers in the dataset are making RMSE around 50% larger than MAE

```
x = sm.add_constant(diamonds[["carat", "depth", "table", "x", "y"]])
y = diamonds["price"]

model = sm.OLS(y, x).fit()

print(f"MAE: {mae(model.predict(x), y)}")
print(f"RMSE: {rmse(model.predict(x), y, squared=False)}")
```

```
MAE: 889.2135691786077
RMSE: 1496.8311707830426
```

RMSE will always be bigger than MAE

The multiple linear regression model performs better across both metrics

ASSIGNMENT: MEAN ERROR METRICS

NEW MESSAGE

July 3, 2023

From: **Cam Correlation** *(Sr. Data Scientist)*

Subject: **Error Metrics**

Hi there!

Thanks for building those models, they're definitely showing some potential!

Can you calculate MAE and RMSE for the model with all variables and our simple regression model that just included RAM?

This will help me communicate the improvement in fit better to our stakeholders.

Thanks!

03_multiple_regression_assignments.ipynb

Reply Forward

Key Objectives

1. Calculate MAE and RMSE for fitted regression models

KEY TAKEAWAYS

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

- *Each new feature comes with an associated coefficient that forms part of the regression equation*

Variable selection methods help you identify valuable features for the model

- *Coefficients with p-values greater than alpha (0.05) indicate that a coefficient isn't significantly different than 0*
- *Contrary to R^2, the adjusted R^2 metric penalizes new variables added to a model*

Mean error metrics let you compare predictive accuracy across models

- *Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) quantify a model's inaccuracy in the target's units*
- *RMSE is more sensitive to large errors, so it is preferred in situations where large errors are undesirable*