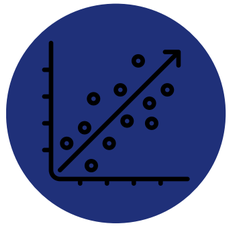
# LIEANR REGRESSION

Regression Reading

# REGRESSION

ទីវិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE



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

## TOPICS WE'LL COVER:

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

## GOALS FOR THIS SECTION:

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

# Linear Regression

ទិន្នាស្ថានសាន់ប៊ុយស៍
SUNRISE INSTITUTE

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

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

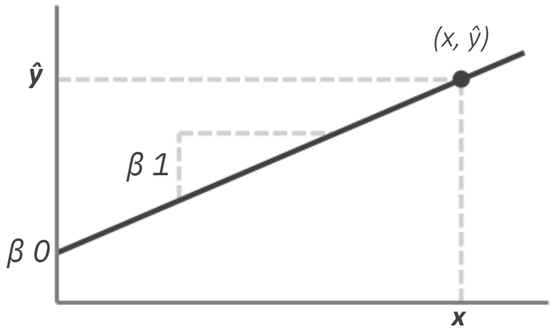
*The **predicted** value for the target*

*The value for the feature*

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

*The **y-intercept***

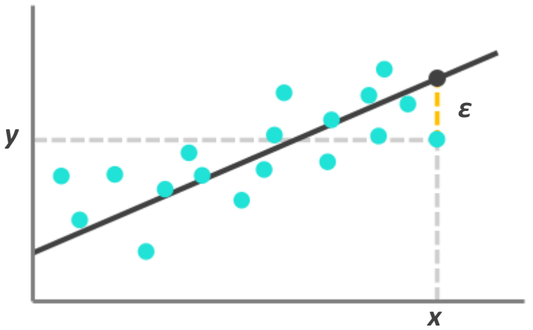
*The **slope** of the relationship*




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

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

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



# Linear Regression





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

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

*SIMPLE LINEAR REGRESSION MODEL:*

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

*MULTIPLE LINEAR REGRESSION MODEL:*

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

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

# GOALS OF Linear REGRESSION

វិទ្យាស្ថានសាន់ប៊ុយស៍
SUNRISE INSTITUTE

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

**LinearRegression**

**Linear with Python**

**Optimization**

**Implementation**

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

## PREDICTION

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

## INFERENCE

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

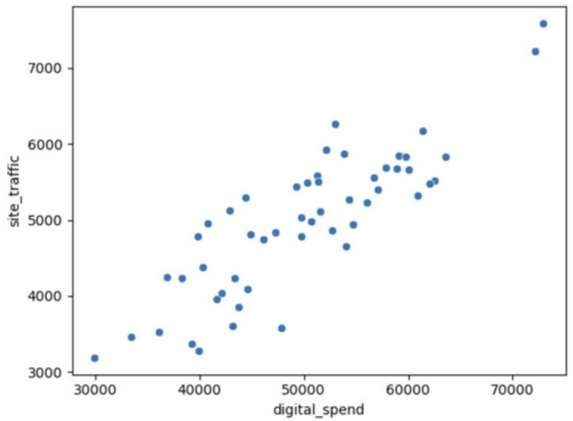
# LINEAR REGRESSION



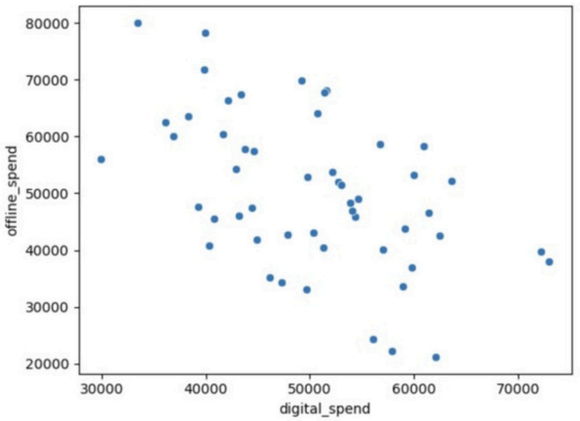
The **correlation coefficient** ‘r’ is single number that shows the linear relationship between two variables.

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

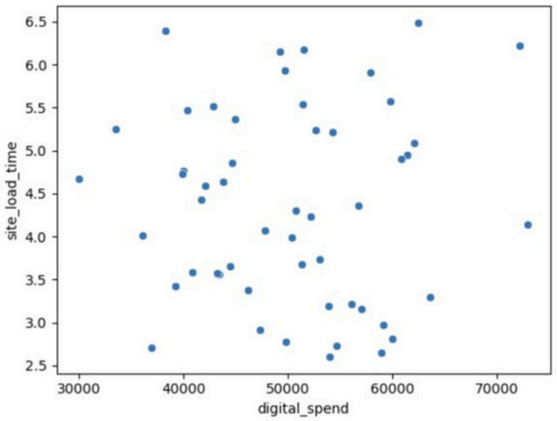
**r = 0.858**

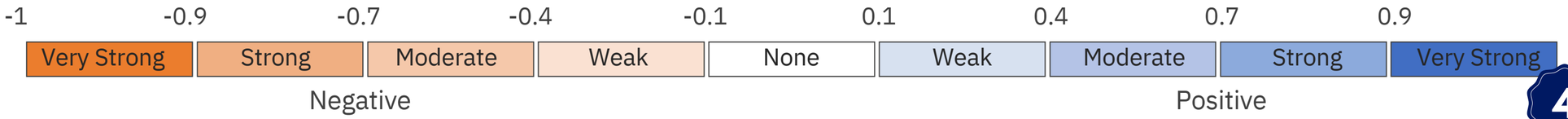



**r= -0.499**




**r= 0.008**



# Regression Metrics

# Model evaluation (Metric) -Regression



Some common regression metrics in scikit-learn

with examples

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

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

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

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

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

# Model evaluation (Metric) -Regression

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

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

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

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

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

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

# IMPLEMENTATION



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

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

statsmodels

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

learn

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

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

# REGRESSION IN STATSMODELS





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

```
import statsmodels.api as sm

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

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

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

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

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

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

## Why do we need to add a constant?

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

# REGRESSION IN STATSMODELS



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

**Regression**

**Linear with Python**

**Optimization**

**Implementation**

```
import statsmodels.api as sm

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

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

OLS Regression Results

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

*Model summary statistics*

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

*Variable summary statistics*

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

*Residual (error) statistics*

# INTERPRETING THE MODEL

ទីវិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

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

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

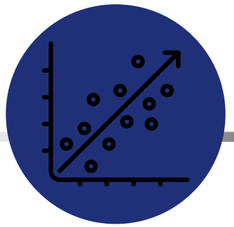
```
import statsmodels.api as sm

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

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

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

**How do we interpret this?**

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



# R-SQUARED

**Regression**

**Linear with Python**

**Optimization**

**Implementation**

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

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

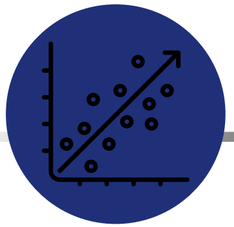
```
import statsmodels.api as sm

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

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

OLS Regression Results

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

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



# R-SQUARED

**Regression**

**Linear with Python**

**Optimization**

**Implementation**

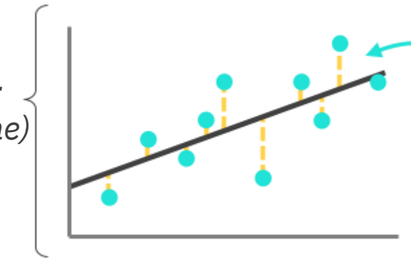
**R-squared**, or coefficient of determination, measures how much better the model is at predicting the target than using its mean *(our best guess without using features)*

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

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

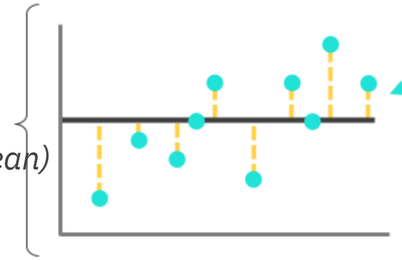
*This is the **sum of squared error** (distance between values & line)*

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



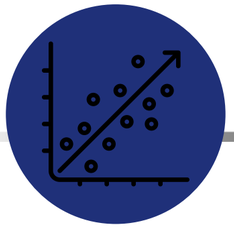
*This is the variation of "y" not explained by "x"*

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



*This is the variation of "y" around its mean*

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



# R-SQUARED

Regression

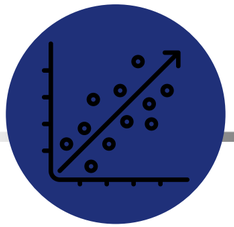
Linear with Python

Optimization

Implementation

## Adjusted $R^2$

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



# HYPOTHESIS TEST

Regression

Linear with Python

Optimization

Implementation

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

# MULTIPLE LIEANR REGRESSION

# Multiple linear regression scenarios
# Goals

# 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 Seletion**

**Mean Error Metric**

## 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 Regression

Variable Selection

Evaluation Metrics

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

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

***SIMPLE LINEAR REGRESSION MODEL:***

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

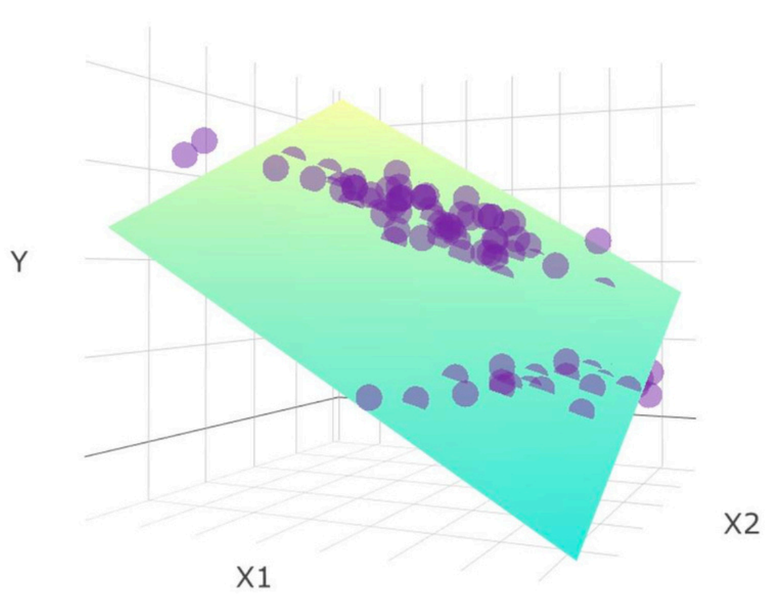
***MULTIPLE LINEAR REGRESSION MODEL:***

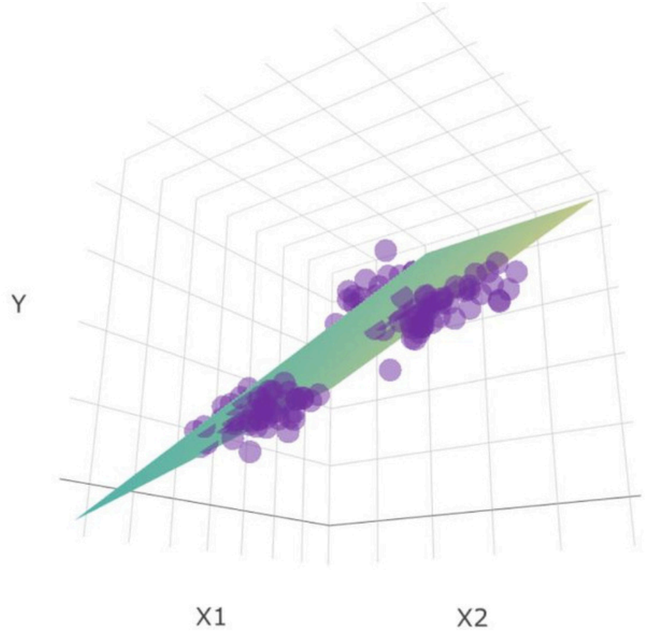
$$y_i = \underbrace{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} + \epsilon}$$

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

# MULTIPLE LINEAR REGRESSION MODEL

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

Variable Selection

Mean Error Metrics

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

## To **fit a multiple regression**

**Multiple Regression**

**Variable Selection**

**Evaluation Metrics**

```
from sklearn.model_selection import train_test_split
```

```
x = df[['age']]
y = df['charges']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=90)
```

```
df.corr()
```

| | age | sex | bmi | children | smoker | charges | region_northeast | region_northwest | region_southeast | region_southwest |
|---|---|---|---|---|---|---|---|---|---|---|
| **age** | 1.000000 | 0.018326 | 0.128016 | 0.032763 | -0.012625 | 0.955779 | 0.014151 | 0.003251 | -0.047004 | 0.029025 |
| **sex** | 0.018326 | 1.000000 | -0.001841 | -0.006287 | -0.001524 | 0.072101 | -0.008953 | -0.000319 | -0.002020 | 0.010989 |
| **bmi** | 0.128016 | -0.001841 | 1.000000 | 0.000765 | -0.087787 | 0.108261 | -0.135730 | -0.120017 | 0.255862 | 0.000401 |
| **children** | 0.032763 | -0.006287 | 0.000765 | 1.000000 | -0.002767 | 0.171814 | -0.027998 | 0.009706 | -0.016625 | 0.033845 |
| **smoker** | -0.012625 | -0.001524 | -0.087787 | -0.002767 | 1.000000 | 0.107042 | 0.027877 | -0.026673 | -0.026013 | 0.024803 |
| **charges** | 0.955779 | 0.072101 | 0.108261 | 0.171814 | 0.107042 | 1.000000 | 0.070417 | 0.010897 | -0.085157 | 0.004127 |
| **region_northeast** | 0.014151 | -0.008953 | -0.135730 | -0.027998 | 0.027877 | 0.070417 | 1.000000 | -0.327673 | -0.319571 | -0.333977 |
| **region_northwest** | 0.003251 | -0.000319 | -0.120017 | 0.009706 | -0.026673 | 0.010897 | -0.327673 | 1.000000 | -0.332356 | -0.347338 |
| **region_southeast** | -0.047004 | -0.002020 | 0.255862 | -0.016625 | -0.026013 | -0.085157 | -0.319571 | -0.332356 | 1.000000 | -0.338750 |
| **region_southwest** | 0.029025 | 0.010989 | 0.000401 | 0.033845 | 0.024803 | 0.004127 | -0.333977 | -0.347338 | -0.338750 | 1.000000 |

*"age" are the most correlated features with "charges"*

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 specifiec

*R2 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

**Evaluation 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

Evaluation 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 |

*R2 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 R2 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

**Evaluation 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:** | 850.1 |
| **Covariance Type:** | nonrobust | **Prob (F-statistic):** | 1.73e-77 |

| | 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 |
| **Covariance Type:** | nonrobust | **Prob (F-statistic):** | 1.73e-77 |

*R2 increased but adjusted R2 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 R2)*

# MEAN ERROR METRICS

Multiple Regression

Variable Selection

**Evaluation 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*

## MSE

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

## RMSE

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

**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

**Evaluation 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*