{desafío}
latam_

Regresión Lineal_

Sesión Presencial 2

Activación de Conceptos

- En la unidad anterior aprendimos sobre la regresión desde la econometría.
- ¡Pongamos a prueba nuestros conocimientos!

¿Cómo implementaríamos la siguiente ecuación de la recta con `statsmodels`?

$$y_i = \beta_0 + \beta_1 \cdot X1 + \beta_2 \cdot X2 + \beta_3 \cdot X3 + \beta_4 \cdot X4 + \varepsilon_i$$

1. `lm('y ~ x1 + x2 + x3 + x4')`

- `smf.ols('y ~ x1 + x2 + x3 + x4')`
- `smf.regress(''y ~ x1 + x2 + x3 + x4')`
- `smf.ols('y ~ x1 + x2 + x3 + x4', df)`

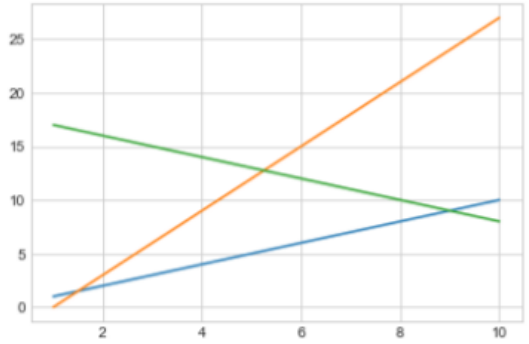
¿Qué significa el intercepto?

1. El valor promedio de la muestra.

- El valor estimado de nuestra variable dependiente cuando $X = 0$.
- El valor estimado de nuestra variable independiente cuando $X = 0$.
- El valore estimado de nuestra variable dependiente cuando $Y = 0$.

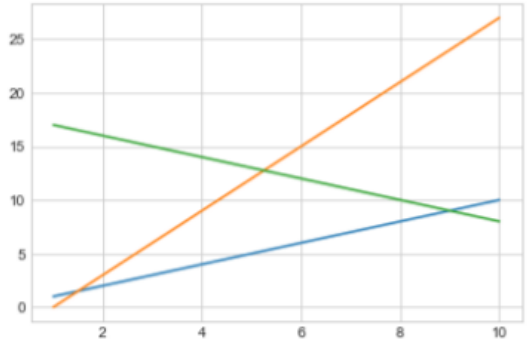
A continuación se grafican 3 ecuaciones de la recta, ¿Cuál presenta una pendiente negativa?

In [17]:
```
plot_regress()
```



A continuación se grafican 3 ecuaciones de la recta, ¿Cuál presenta una pendiente más pronunciada?

In [18]:
```
plot_regress()
```



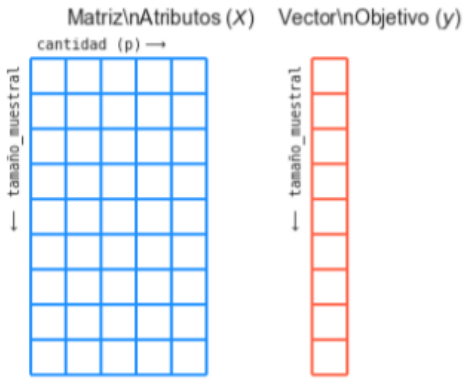
Regresión Lineal (Desde Machine Learning)

{desafío}
latam_

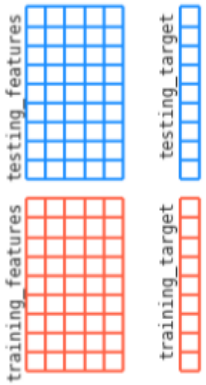
Dividir

- El primer paso es dividir nuestra muestra en dos grupos:
  - **Entrenamiento:** de donde generamos una aproximación funcional.
  - **Validación:** donde ponemos a prueba nuestra aproximación funcional a nuevos datos.

In [20]:
```
gfx.feature_target()
```



In [21]:
```
gfx.train_testing()
```



Importando módulos

- Para realizar machine learning utilizaremos `scikit-learn`, una librería con una amplia gama de funciones y modelos para simplificar el flujo de trabajo.

In [22]:
```
# importamos train_test_split
from sklearn.model_selection import train_test_split
```

In [23]:
```
# separemos los vectores a trabajar
y_vec= df.loc[:, 'earn']
X_mat = df.loc[:, 'height1':'male']
```

In [24]:
```
# dividimos la muestra en entrenamiento y validación
X_train, X_test, y_train, y_test = train_test_split(X_mat, y_vec, test_size=.30, random_s
tate=11238)
```

Entrenar modelo

- El primer paso es definir un algoritmo a utilizar. En este caso implementaremos `LinearRegression`, donde especificaremos que deseamos incluir un parámetro para el intercepto y deseamos normalizar las variables.

In [25]:
```
from sklearn import linear_model
model_1 = linear_model.LinearRegression(fit_intercept=True, normalize=True)
```

- Una vez inicializado el objeto con nuestras especificaciones, podemos entrenar el modelo con `.fit`

In [26]:
```
model_1.fit(X_train, y_train)
```

Out[26]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=
True)
```

- El último paso es generar los puntajes predichos ($\hat{y}$) con `.predict`

In [27]:
```
model_1_yhat = model_1.predict(X_test)
```

Medir desempeño

- El desempeño de un modelo se comprueba mediante las métricas.
- Para el caso de la regresión lineal, nuestro objetivo es encontrar una función candidata que minimice las pérdidas, medidas en el Error Cuadrático Promedio

$$\text{MSE}(\hat{f}, \text{datos}) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}(\mathbf{x_i}) \right)^2$$

- Para ello debemos importar funciones ya incorporadas en `sklearn`.

In [28]:
```
from sklearn.metrics import mean_squared_error, r2_score
```

- Con las funciones importadas podemos evaluar qué tan bien se desempeñó.

In [29]:
```
print(mean_squared_error(y_test, model_1_yhat))
print(r2_score(y_test, model_1_yhat))
```

```
246103243.8560099
0.1718607956610353
```

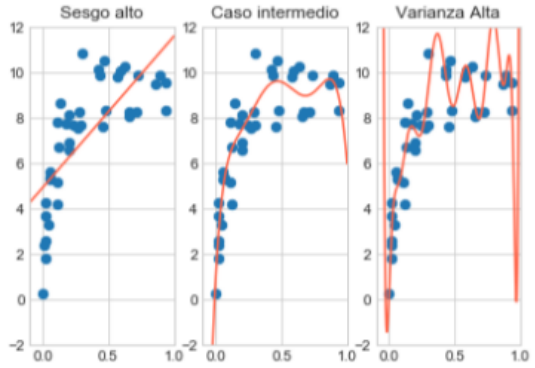
Teoría del Aprendizaje 101

{desafío}
latam_

Sesgo y Varianza

- División de la muestra → **aumentar la capacidad de generalización del modelo**

In [30]:
```
gfx.bias_variance()
```



El objetivo del aprendizaje supervisado es especificar los parámetros de forma tal de encontrar un modelo que sea lo suficientemente flexible con el caso estudiado, pero que también pueda ser extrapolado a nuevos datos y tener un buen desempeño.

Validación

- Como se comportan las métricas de desempeño en la medida que aumentamos la complejidad paramétrica del modelo.

In [31]:
```
gfx.validate_curve()
```

Aprendizaje

- Como se comportan las métricas de desempeño en la medida que el tamaño muestral del entrenamiento $n \to \infty$.

In [32]:
```
gfx.learning_curve()
```