Análisis de datos

Estadística

Transformación Digital
Agencia de Transformación Digital y Telecomunicaciones

TECNOLOGICO NACIONAL DE MEXICO®

La estadística es esencial para analizar datos. Se utiliza en todas las fases del análisis de datos; desde el preprocesamiento (limpieza, reducción), hasta el desarrollo y monitoreo de modelos de aprendizaje automático. Los métodos estadísticos pueden emplearse para resumir o describir un conjunto de datos. La estadística también es útil para extraer diversos patrones de los datos, así como para comprender los mecanismos que los generan y afectan (Han et al., 2023).

La estadística es la práctica de recopilar y analizar datos para descubrir hallazgos útiles o predecir las causas de dichos hallazgos. La probabilidad suele desempeñar un papel importante en la estadística, ya que utilizamos datos para estimar la probabilidad de que ocurra un evento.

Puede que no siempre se le reconozca, pero la estadística es el núcleo de muchas innovaciones basadas en datos. El aprendizaje automático en sí mismo, es una herramienta estadística que busca posibles hipótesis, para correlacionar vínculos lógicos entre diferentes variables en el universo de los datos.

Estadística descriptiva

Es esencial tener una imagen general de los datos cuando se preparan para su análisis. Las técnicas de resumen de datos descriptivo pueden usarse para identificar las propiedades típicas de los datos y seleccionar cuáles valores de datos deben tratarse como ruido o atípicos, es decir, valores que se desvían significativamente del resto de los datos.

En esta sección se presentan los conceptos básicos del resumen de datos descriptivo. Esta sección inicia con las *medidas de tendencia central*, que miden la ubicación de la mitad o del centro en una distribución de datos.

Intuitivamente hablando, dado un atributo, ¿dónde caen la mayoría de sus valores? En particular, se discuten la *media*, la *mediana* y la *moda*.

Además de evaluar la tendencia central del conjunto de datos, también es importante tener una idea de la *dispersión de los datos*. Es decir, ¿cómo se extienden los datos? Las medidas de dispersión de datos más comunes, son la *varianza* y la *desviación estándar* de los datos (Han et al., 2023).

Medidas de tendencia central

Media

La *media* es el promedio de un conjunto de valores. La operación es sencilla: sumar los valores y dividirlos entre el número de valores. La media es útil porque muestra dónde se encuentra el "centro de gravedad" de un conjunto de valores observados.

La media se calcula de la misma manera, tanto para poblaciones como para muestras. La Figura 3.1 ilustra una muestra de ocho valores y cómo calcular su media en Python utilizando el módulo *statistics,* que proporciona funciones para calcular estadísticas matemáticas de datos numéricos (Python, 2024).

Figura 3.1

```
1 import statistics as st
2 # Número de mascotas que cada persona tiene
3 muestra = [1, 3, 2, 5, 7, 0, 2, 3]
4 print(st.mean(muestra)) # imprime 2.875
```

Cálculo de la media en Python.

Como se observa en la Figura 3.1, se encuestó a ocho personas sobre el número de mascotas que tienen. La suma de la muestra es 23 y el número de elementos en la muestra es 8, lo que nos da una media de 2.875, ya que 23/8 = 2.875 (Nield, 2022).

Aunque la media es la cantidad útil más simple para describir un conjunto de datos, no siempre es la mejor forma de medir el centro de los mismos. Un problema principal con la media, es su sensibilidad a los valores extremos (por ejemplo, los atípicos). Incluso un pequeño número de valores extremos puede corromper (dañar) la media. Por ejemplo, la media del salario en una compañía, puede incrementarse sustancialmente debido al monto que se le paga a unos cuantos gerentes que ganan mucho. De igual forma, la calificación promedio de una clase en un examen puede bajar considerablemente, debido a unas pocas calificaciones bajas (Han et al., 2023).

Mediana

Para datos sesgados (con una distribución asimétrica), una mejor medida del centro de los datos es la *mediana*. Suponiendo que un conjunto determinado de datos, con *N* valores distintos, se ordena numéricamente. Si *N* es impar, entonces la mediana es el valor medio del conjunto ordenado; si *N* es par, la mediana es el promedio de los dos valores intermedios (Han et al., 2023). Puede observarse en la Figura 3.2 que la mediana del número de mascotas de la muestra es 7.

Figura 3.2

```
1 import statistics as st
2 # Número de mascotas que cada persona tiene
3 muestra = [0, 1, 5, 7, 9, 10, 14]
4 print(mediana(muestra)) # imprime 7
```

Cálculo de la mediana en Python.

Moda

La *moda* para un conjunto de datos, es el valor que ocurre con mayor frecuencia en el conjunto. Es posible que la frecuencia más grande corresponda a varios valores diferentes, lo cual resulta en más de una moda. Los conjuntos de datos con una, dos o tres *modas* se llaman unimodales, bimodales y trimodales, respectivamente. En general, un conjunto de datos con una o más modas es *multimodal*. En el otro extremo, si cada valor de datos ocurre solo una vez, entonces no hay moda (Han et al., 2023). El conjunto de datos de las mascotas es bimodal porque tanto 2 como 3 ocurren con mayor frecuencia. En la Figura 3.3 se calcula la moda para el ejemplo.

Figura 3.3

```
1 import statistics as st
2 # Número de mascotas que cada persona tiene
3 muestra = [1, 3, 2, 5, 7, 0, 2, 3]
4 print(st.multimode(muestra)) # imprime [3,2]
```

Cálculo de la moda en Python.

Medidas de dispersión

Estas medidas son útiles para identificar atípicos (que se desvían del resto de los datos). La dispersión es el grado en el que los datos numéricos tienden a propagarse. También se le conoce como *varianza de los datos*.

La varianza de *N* observaciones, x_1, x_2,...,x_N, es

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \left(\frac{1}{N}\sum_{i=1}^{N}x_i^2\right) - \bar{x}^2$$

Donde $\bar{x}$ es el valor de la media de las observaciones, como se define en la ecuación. La desviación estándar (σ) de las observaciones es la raíz cuadrada de la varianza (σ^2).

En la Figura 3.4 se obtiene la varianza para valores de *salario* (en miles de pesos) ordenados ascendentemente.

Figura 3.4

```
1 import statistics as st
2 # Valores para salario
3 muestra = [30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70, 110]
4 print(st.pvariance(muestra)) # imprime 379.17
```

Cálculo de la varianza en Python.

Las propiedades básicas de la desviación estándar (σ) como una medida de dispersión son (Han et al., 2023):

- σ mide la dispersión sobre la media y debe usarse solo cuando la media se elige como medida central.

- $\sigma=0$ solo cuando no hay dispersión, es decir, cuando todas las observaciones tienen el mismo valor. De otra forma $\sigma>0$.

Estadística inferencial

La estadística inferencial intenta descubrir atributos sobre una población más grande, frecuentemente basándose en una muestra. Generalmente se malinterpreta y es menos intuitiva que la estadística descriptiva. A menudo nos interesa estudiar un grupo demasiado grande, para observar sus características (por ejemplo, la estatura promedio de los adolescentes en México), y tenemos que recurrir a usar solo unos pocos miembros de ese grupo para inferir conclusiones sobre ellos. Como puedes imaginar, esto no es fácil de acertar. Después de todo, intentamos *representar* una población, con una muestra que puede no ser representativa (Nield, 2022).

Correlación y regresión lineal simple

Una de las técnicas más prácticas en el análisis de datos, es ajustar una línea a través de puntos de datos observados, para mostrar una relación entre dos o más variables. Una regresión es el intento de ajustar una función (una regla matemática que relaciona dos variables) a los datos observados, para hacer predicciones sobre nuevos datos. Una *regresión lineal* ajusta una línea a los datos observados, intentando demostrar una relación lineal entre variables y hacer predicciones sobre nuevos datos aún por observar.

La regresión lineal es un caballo de batalla de la minería de datos y la estadística. Esta técnica, relativamente simple, ha existido por más de doscientos años y, actualmente, se considera una forma de aprendizaje automático.

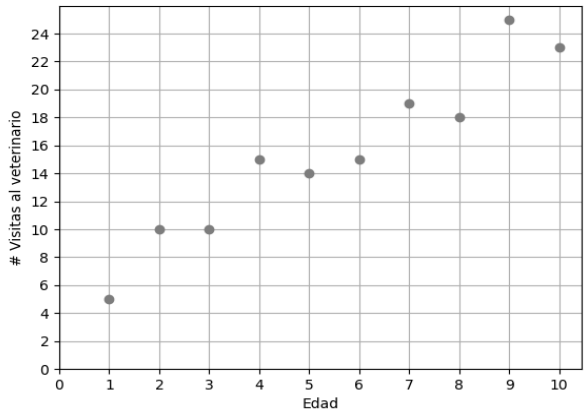
Por ejemplo: se quiere estudiar la relación entre la edad de un perro y el número de veterinarios que ha visitado. En una muestra fabricada se tienen 10 perros aleatorios. Se grafica este conjunto de datos, como se muestra en la Figura 3.5.

Claramente se observa que existe una *correlación lineal*, esto significa que cuando una de estas variables incrementa/decrementa, la otra incrementa/decrementa en una cantidad aproximadamente proporcional. Es posible dibujar una línea a través de estos puntos para mostrar una correlación con esta en la Figura 3.6. El principal beneficio de la regresión lineal es que permite hacer predicciones en los datos que no se han observado antes. No se tiene un perro en la muestra que tenga 8.5 años, pero es posible ver esta línea y estimar que el perro habría visitado 21 veces al veterinario en su vida. Solo se observa la línea donde x=8.5 y se ve que y=21.218 como se muestra en la Figura 3.7. Otro beneficio es que proporciona la capacidad de analizar variables para encontrar relaciones posibles y suponer que las variables correlacionadas son causales entre ellas.

Sin embargo, no es posible esperar que cada resultado se ubique exactamente en esa línea. Después de todo, los datos del mundo real son ruidosos y nunca perfectos y no seguirán una línea recta. En ese caso, existirá error alrededor de esa línea, cuando el punto aparece sobre o bajo la misma. Por lo tanto, es necesario obtener valores p (que indican la

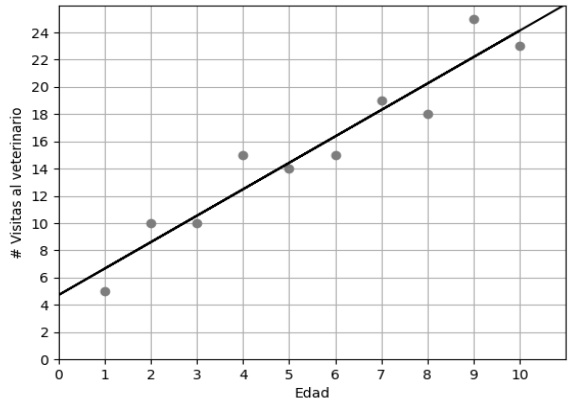
probabilidad de que algo ocurra por casualidad), *significancia estadística* (indicador que nos ayuda a determinar si un resultado obtenido en un análisis es poco probable que haya ocurrido por azar) e *intervalos de predicción* (rango dentro del cual se espera que caiga un valor futuro de una variable), que describen qué tan confiable es la regresión lineal. Otro problema es que no se debería usar la regresión lineal para hacer predicciones fuera del rango de los datos que se tienen, esto significa que en este ejemplo no se deberían hacer predicciones donde $x < 0$ y $x > 10$ porque no se tienen datos fuera de esos valores (Nield, 2022).

Figura 3.5



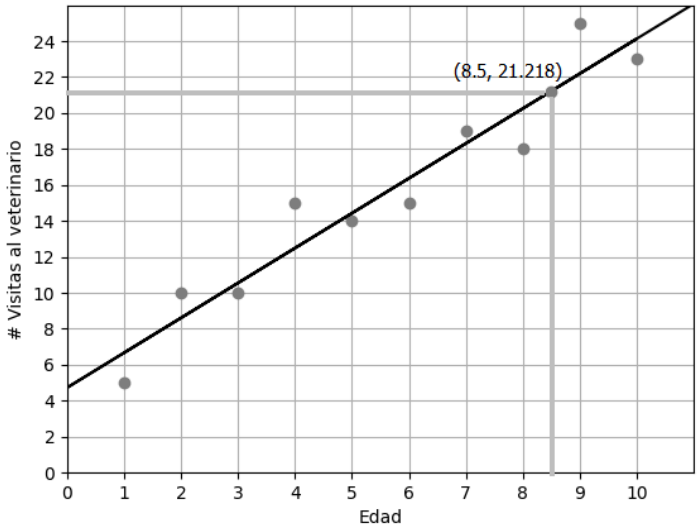
Gráfica de una muestra de 10 perros con su edad y número de visitas al veterinario (Nield, 2022).

Figura 3.6



Ajuste de línea a través de los datos (Nield, 2022).

Figura 3.7



Predicción usando una regresión lineal, se muestra que un perro de 8.5 años visitará aproximadamente 21.2 veces al veterinario (Nield, 2022).

Regresión lineal con Scikit-learn

La Figura 3.8 muestra cómo se usa *scikit-learn* (biblioteca de aprendizaje automático de código abierto que soporta la construcción de modelos predictivos y descriptivos) para realizar una regresión lineal básica sobre la muestra de 10 perros. Los datos se extraen utilizando Pandas (líneas 7 a 14), se ejecuta la regresión lineal usando *scikit-learn* (líneas 20 a 27) y se muestra en un gráfico (líneas 30 a 40).

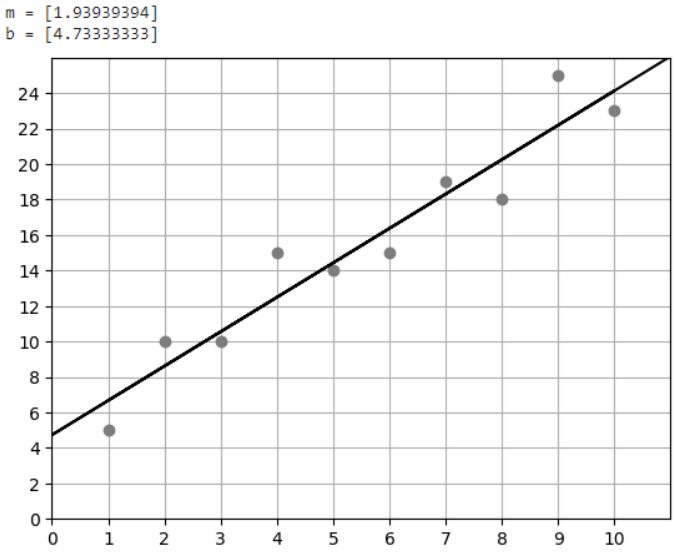
En la línea 7, se importan los datos de este CSV en GitHub (*https://bit.ly/2KF29Bd*). Se separan las dos columnas en los conjuntos de datos *edad* y *visitas* usando Pandas (líneas 10 y 14). Entonces, se ajusta con el método *fit()* el modelo de regresión lineal (*LinearRegression*) a los datos de entrada *edad* y los datos de salida *visitas* (línea 21). Después, se obtienen los coeficientes *m* (pendiente) y *b* (intersección) que describen la función lineal ajustada (líneas 24 a 27). En la gráfica de la Figura 3.9, se observa una línea ajustada a estos puntos y los valores de la pendiente y la intersección (Nield, 2022).

Figura 3.8

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import numpy as np

# Importar datos
df = pd.read_csv('https://bit.ly/2KF29Bd', delimiter=",")

# Extraer las variables de entrada (todas las filas, última columna)
visitas = df.values[:, -1]

# Extraer la columna de salida (todas las filas, todas las columnas menos
# la última)
edad = df.values[:, :-1]

edad = edad.reshape(-1,1)
visitas=visitas.reshape(-1,1)

#Ajustar una línea a los puntos
model=LinearRegression()
model.fit(edad, visitas)

# m = 1.7867224, b = -16.51923513
m=model.coef_.flatten()
b=model.intercept_.flatten()
print("m = {0}".format(m))
print("b = {0}".format(b))

# Gráfico
plt.grid(True)
plt.yticks(range(0,26,2))
plt.xticks(range(0,11,1))
plt.ylim(0, 26)
plt.xlim(0, 11)
plt.plot(edad, visitas, 'o', color='gray') # gráfico de dispersión
edad2 = df['x'].to_numpy()
edad3 = np.append(edad2, [0, 12])
plt.plot(edad3, m*edad3+b, color='black') # línea
plt.savefig('diagrama-dispersion5.png')
plt.show()
```

Uso de scikit-learn para realizar una regresión lineal (Nield, 2022)

Figura 3.9



Línea de regresión obtenida con scikit-learn (Nield, 2022)

Residuos de errores cuadráticos

¿Cómo crean las herramientas estadísticas una línea que se ajuste a estos puntos? Todo se reduce a dos preguntas que son fundamentales para el entrenamiento de aprendizaje automático (*machine learning*):

- ¿Qué define un “mejor ajuste”?
- ¿Cómo se obtiene ese “mejor ajuste”?

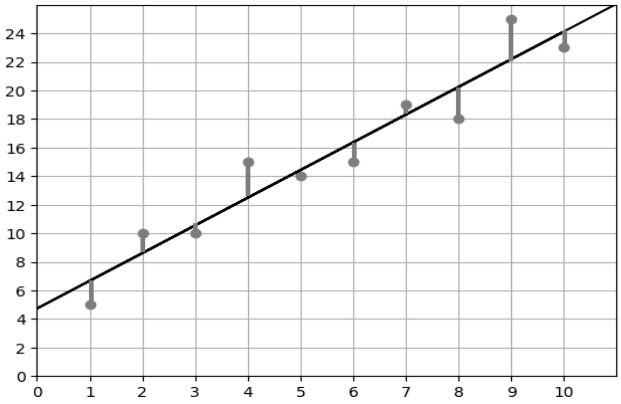
La primera pregunta tiene una respuesta bastante establecida: se minimizan los cuadrados, o más específicamente, la suma de los residuos cuadráticos. El residuo es la diferencia numérica entre la línea y los puntos, como se muestra en la Figura 3.10.

Los puntos sobre la línea tendrán un *residuo positivo*, y los puntos debajo de la línea tendrán un *residuo negativo*. En otras palabras, el residuo es la diferencia restada entre los valores de *y* predichos (derivados de la línea) y

los valores de *y* reales (que provienen de los datos). Otro nombre para los residuos son *errores*, porque ellos reflejan qué tan errónea es la línea para predecir los datos.

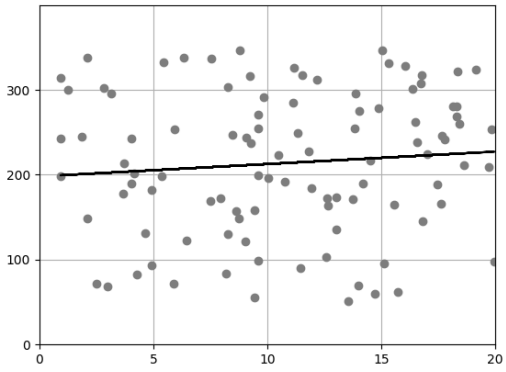
Observe el diagrama de dispersión de la Figura 3.11 junto con su regresión lineal. ¿Por qué parece que la regresión lineal no funciona muy bien aquí?

Figura 3.10



Los residuos son las diferencias entre la línea y los puntos (Nield, 2022).

Figura 3.11



Un gráfico de dispersión de los datos con alta varianza (Nield, 2022).

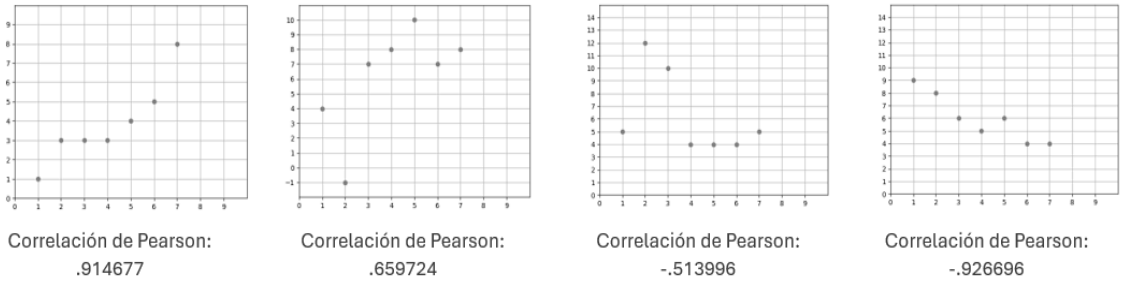
El problema aquí es que los datos tienen alta varianza. Si los datos están demasiado dispersos, la varianza subirá hasta el punto en que las predicciones serán menos exactas y útiles, resultando en residuos grandes. Además, el desajuste también debilitará las predicciones porque los datos están extendidos. Se requiere medir numéricamente, qué tan fuera de lugar están las predicciones.

Entonces, ¿cómo se miden estos residuos en conjunto? ¿cómo se obtiene también un sentido de qué tan mala es la varianza en los datos? Para esto se utiliza el coeficiente de correlación, también llamado *correlación de Pearson*, que mide la fuerza de la relación entre dos variables entre -1 y 1. Un coeficiente de correlación cercano a 0 indica que no hay correlación, mientras que cuando se acerca a 1 indica una correlación positiva, esto significa que, si una variable incrementa, la otra también incrementa proporcionalmente. Si es cercano a -1, entonces indica una correlación negativa fuerte, que significa que mientras una variable incrementa la otra decrementa proporcionalmente.

El coeficiente de correlación se denota como *r*. Los datos dispersos de la Figura 3.11 tienen un coeficiente de correlación de 0.093. Debido a que es mucho más cercano a 0 que a 1, por lo tanto, es posible inferir que los datos tienen poca correlación.

Los cuatro diagramas de dispersión de la Figura 3.12 muestran sus coeficientes de correlación. Mientras los puntos sigan más una línea, la correlación es más fuerte. Puntos más dispersos resultan en correlaciones más débiles.

Figura 3.12



Coeficientes de correlación para cuatro gráficos de dispersión (Nield, 2022).

Como es posible imaginar, el coeficiente de correlación es útil para ver si hay una relación posible entre dos variables. Si hay una relación positiva-negativa fuerte, serán útiles en la regresión lineal. Si no hay una relación, solo agregarían ruido y dañarían la precisión del modelo (Nield, 2022).

¿Cómo se usa Python para calcular el coeficiente de correlación? Se usa el conjunto de datos simple de 10 puntos *(https://bit.ly/2KF29Bd*) que se utilizó anteriormente. Una forma rápida y fácil de analizar correlaciones para todos los pares de variables es usar la función de Pandas corr(). Esta facilita ver el coeficiente de correlación entre cada par de variables en un conjunto de datos, que en este caso solo son *x* y *y*. Esta se conoce como *matriz de correlación* (tabla que muestra el grado de relación entre varias variables al mismo tiempo). El código se observa en la Figura 3.13 y la matriz de correlación en la Figura 3.14.

Figura 3.13

```
import pandas as pd

# Leer los datos en el dataframe de Pandas
df=pd.read_csv('https://bit.ly/2KF29Bd', delimiter=",")

# Imprimir correlaciones entre variables
correlaciones=df.corr(method='pearson')
print(correlaciones)
```

Uso de Pandas para ver el coeficiente de correlación entre cada par de variables (Nield, 2022).

Figura 3.14

```
          x         y
x  1.000000  0.957586
y  0.957586  1.000000
```

Matriz de correlación que resulta de la ejecución del código de la Figura 3.13 (Nield, 2022).

Como se observa, el coeficiente de correlación 0.957586 entre *x* y *y* indica una correlación positiva fuerte entre las dos variables. Es posible ignorar las partes de la matriz donde se observa las correlaciones de *x* y *y* consigo mismas que tienen un valor de 1.0. En estos casos la correlación será perfecta de 1.0, porque los valores corresponden consigo mismos exactamente (Nield, 2022).

Elaboró contenido: Dra. Lisbeth Rodríguez Mazahua

Referencias:

- Desmos. (2025). *desmos*. Beta Distribution. https://www.desmos.com/calculator/xylhmcwo71?lang=es
- Han, J., Pei, J., & Tong, H. (2023). *Data Mining: Concepts and Techniques* (Fourth Edition). Morgan Kaufmann.
- Nield, T. (2022). *Essential Math for Data Science*. O'Reilly Media, Inc.
- Python. (2024). *statistics — Funciones de estadística matemática — documentación de Python - 3.8.20*. https://docs.python.org/es/3.8/library/statistics.html
- VentureBeat. (2019). *Why Do 87% of Data Science Projects Never Make It into Production?* Venture Beat. https://venturebeat.com/ai/why-do-87-of-data-science-projects-never-make-it-into-production