INDICE

Estadísticas

- **Tipos**
    - **Conceptos**
    - **Casos de uso en machine learning**

Tipos de datos

- **Numericos**
    - **Variables discretas**
    - **Variables continuas**

- **Categoricos**
    - **Variables ordinales**
    - **Variables nominales**

Funcion describe

- **Concepto**
- **Definición de sus métodos estadísticos**
    - **Count**
    - **Mean**
    - **Std (Desviación estándar)**
    - **Min**
    - **Max**

Sesgo y varianza

- **Conceptos**

Tipos de errores

- Conceptos
- Tipos
- Casos de uso

Estadistica

Estadística descriptiva:

Ofrece métodos para resumir los datos, transformando las informaciones recolectadas en bruto en información significativa. Es decir, se trata de un conjunto de técnicas y herramientas que nos permiten analizar y presentar los datos de forma resumida y comprensible.

Casos de uso

Algunas aplicaciones de la estadística descriptiva son:

- Describir y resumir características de un conjunto de datos, como su media, mediana, moda y dispersión.
- Identificar patrones o tendencias en los datos.
- Realizar comparaciones entre diferentes grupos de datos o poblaciones.
- Detectar valores atípicos o datos faltantes.
- Visualizar los datos utilizando gráficos como histogramas, boxplots, gráficos de dispersión, entre otros.

Ejemplos en python

Podemos aplicar la estadística descriptiva a una dataframe utilizando las funciones estadísticas de la librería de pandas.

Cálculo de medidas de tendencia central: Pandas permite calcular fácilmente medidas de tendencia central como la media, la mediana y la moda de un conjunto de datos utilizando los métodos `mean()`,`median()` y `mode()` respectivamente.

```
import pandas as pd

# Crear un DataFrame con datos aleatorios
df = pd.DataFrame({'A': [1, 2, 3, 4, 5], 'B': [2, 4, 6, 8, 10]})
```

Calcular la media de las columnas
→ print(df.mean())
Calcular la mediana de las columnas
→ print(df.median())
Calcular la moda de las columnas
→ print(df.mode())

Cálculo de medidas de dispersión: Pandas también permite calcular medidas de dispersión como la desviación estándar y la varianza utilizando los métodos std() y var(). Por ejemplo:

Calcular la desviación estándar de las columnas
→ print(df.std())

Calcular la varianza de las columnas
→ print(df.var())

Tip: Si solo queremos saber la estadística de una columna en especifico ponernos hacerlos de la manera siguiente:
→ print(df[‘columna'].var())

La estadística inferencial

Ofrece métodos para resumir los datos transformando las observaciones en bruto en información significativa que es fácil de interpretar y compartir. Es decir, la estadística inferencial utiliza métodos estadísticos para analizar un conjunto de datos y hacer inferencias sobre una población más grande basándose en la información contenida en la muestra.

Casos de uso

- Estimar el tamaño de una población a partir de una muestra.
- Realizar pruebas de hipótesis para determinar si dos grupos son estadísticamente diferentes.

- Utilizar modelos estadísticos para predecir el comportamiento futuro de una población o mercado.
- Evaluar el efecto de un tratamiento o intervención en un grupo de pacientes.
- Identificar factores que contribuyen a una enfermedad o condición.

Ejemplos en python

Regresión lineal: La regresión lineal es una herramienta estadística que permite modelar la relación entre dos variables.En python,la librería Scikit-learn ofrece una gran cantidad de herramientas para realizar análisis de regresión lineal, como la función **LinearRegression().**

```
# Crear un DataFrame con datos aleatorios
df = pd.DataFrame({'X': [1, 2, 3, 4, 5], 'Y': [2, 4, 6, 8, 10]})

# Crear un modelo de regresión lineal
model = LinearRegression()

# Entrenar el modelo con los datos
model.fit(df[['X']], df['Y'])

# Realizar una predicción con el modelo
prediction = model.predict([[6]])

# Imprimir la predicción
print("Predicción: ", prediction) [12.]
```

Tipos de datos

Numericos

Los datos numéricos son aquellos que representan cantidades o valores numéricos, y pueden ser continuos o discretos.

Variables discretas

Son aquellos valores que tienen un rango finito.Es decir, que tienen una cantidad numerable de posibles valores, ejemplos:

- Número de hijos: esta variable sólo puede tomar valores enteros, como 0, 1, 2, 3, etc. No se pueden tener 2.5 hijos, por ejemplo.

- Número de estudiantes en una clase: esta variable también es discreta, ya que sólo puede tomar valores enteros.

- Número de goles en un partido de fútbol: esta variable es discreta porque sólo se pueden marcar goles enteros, como 0, 1, 2, 3, etc.

Variables continuas

Las variables continuas son aquellas que pueden tomar cualquier valor dentro de un intervalo o rango determinado, es decir, pueden tomar valores continuos.

- Altura de una persona: la altura puede variar en un rango continuo, por lo que es una variable continua. Una persona puede medir 1,68 metros, 1,68.1 metros, 1,68.2 metros, y así sucesivamente.

- Peso de una persona: el peso también puede variar en un rango continuo, por lo que es una variable continua. Una persona puede pesar 70,3 kilogramos, 70,35 kilogramos, 70,36 kilogramos, etc.

- Temperatura del agua en un lago: la temperatura del agua también puede variar en un rango continuo. Puede medirse en grados Celsius, por ejemplo, y puede tomar valores como 20,5 grados, 20,55 grados, 20,6 grados, y así sucesivamente.

Datos categoricos

Los datos categóricos son aquellos que representan una categoría o una etiqueta, sin un orden o jerarquía preestablecido, estos pueden ser ordinales o nominales.

Variables ordinales

Las variables ordinales son aquellas que representan una categoría con un orden o jerarquía preestablecido.

Nivel de educación: el nivel de educación es una variable ordinal, ya que se puede ordenar jerárquicamente desde el nivel más bajo al más alto. Por ejemplo, primaria, secundaria, bachillerato, universidad.

Clasificación socioeconómica: Se puede ordenar jerárquicamente desde la clase más baja a la más alta. Por ejemplo, baja, media, alta.

Grado de satisfacción: Se puede ordenar jerárquicamente desde el nivel más bajo al más alto. Por ejemplo, insatisfecho, poco satisfecho, satisfecho, muy satisfecho.

Variables nominales

Las variables nominales son aquellas que representan una categoría o etiqueta, pero sin un orden o jerarquía preestablecido.

Color de ojos: el color de ojos es una variable nominal, ya que no hay un orden específico. Por ejemplo, marrón, azul, verde, etc.

Nacionalidad: la nacionalidad es una variable nominal, ya que no hay un orden específico. Por ejemplo, mexicano, estadounidense, canadiense, etc.

Estado civil: el estado civil es una variable nominal, ya que no hay un orden específico. Por ejemplo, soltero, casado, divorciado, viudo, etc.
Uso de la estadística en machine learning:

- Hacer preguntas sobre los datos
- Limpiar y preprocesar los datos
- Seleccionar las características adecuadas

Funcion describe()

Es una función de Pandas que se utiliza para obtener estadísticas descriptivas de un DataFrame o de una Serie de Pandas. Estas estadísticas incluyen la cuenta **(count)**, la media **(mean)**, la desviación estándar **(std)**, el valor mínimo **(min)**, el percentil 25 **(25%)**, la mediana **(50%)**, el percentil 75 **(75%)** y el valor máximo **(max)** de cada columna **numérica** en el DataFrame.

```
import pandas as pd

# Creamos un DataFrame con datos aleatorios
df = pd.DataFrame({'A': [1, 2, 3, 4, 5], 'B': [10, 20, 30, 40, 50]})


# Utilizamos describe() para obtener estadísticas descriptivas
print(df.describe())
```

	A	B
count	5.000000	5.000000
mean	3.000000	30.00000
std	1.581139	15.81139
men	1.000000	10.00000
25%	2.000000	20.00000
50%	3.000000	30.00000
75%	4.000000	40.00000
max	5.000000	50.00000

Como se puede observar, describe() nos proporciona un resumen estadístico de las columnas numéricas en el DataFrame. Esto es muy útil para tener una idea rápida de cómo se distribuyen los datos y detectar posibles valores atípicos o valores faltantes en nuestro conjunto de datos.

En ese mismo orden, la función describe también nos puede ayudar en el análisis de datos categóricos. El parámetro describe(include='all') se utiliza para incluir estadísticas descriptivas de todas las columnas, incluyendo las columnas que contienen datos categóricos.

unique: representa el número de valores únicos que aparecen en la variable categórica.
top: indica la categoría o valor más común en la variable categórica.
freq: indica la frecuencia con la que aparece la categoría más común en la variable categórica.

Por ejemplo, si se tiene una variable categórica llamada "frutas" con los valores "manzana", "pera", "naranja", "manzana", "manzana", "uva", entonces:

unique sería 4, ya que hay 4 valores únicos ("manzana", "pera", "naranja", "uva").
top sería "manzana", ya que es la categoría más común en la variable categórica.
freq sería 3, ya que "manzana" aparece 3 veces en la variable categórica, que es la frecuencia con la que aparece la categoría más común.

Sesgo y Varianza

El sesgo y la varianza son dos conceptos importantes en el aprendizaje automático que están estrechamente relacionados con el rendimiento del modelo.

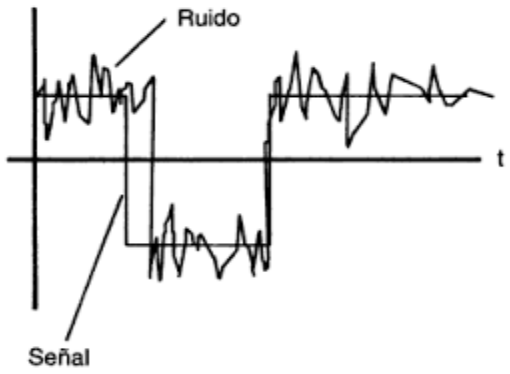
El sesgo se refiere a la tendencia del modelo a hacer predicciones incorrectas en promedio, independientemente de los datos de entrenamiento. Si un modelo tiene un alto sesgo, se dice que está infra-ajustado o sub-ajustado, lo que significa que un modelo no puede capturar la complejidad de los datos de entrenamiento y, por lo tanto, no logra hacer predicciones precisas incluso en los datos de entrenamiento.

La varianza se refiere a la sensibilidad del modelo a las fluctuaciones en los datos de entrenamiento. Si un modelo tiene una alta varianza, se dice que está sobre ajustado, lo que significa que el modelo se ajusta demasiado a los datos de entrenamiento y no puede generalizar bien a nuevos datos.

Vocabulario
Sesgo: Que tan bien o mal un modelo puede predecir un objetivo usando características. Un sesgo menor es mejor.

Subajuste: Otra palabra para sesgo. Un modelo sub ajustado crea malas predicciones en ambos conjuntos de entrenamiento y de prueba.
Varianza: El poder de predicción de un modelo variará en función de los distintos conjuntos de datos. Una varianza baja es mejor.
Sobreajuste: Otro nombre para la varianza. Un modelo sobre ajustado hace buenas predicciones en un conjunto de entrenamiento, pero malas predicciones en un conjunto de prueba.
Ruido: Aleatoriedad natural en los datos.
Señal: La verdadera función de las características a un objetivo.

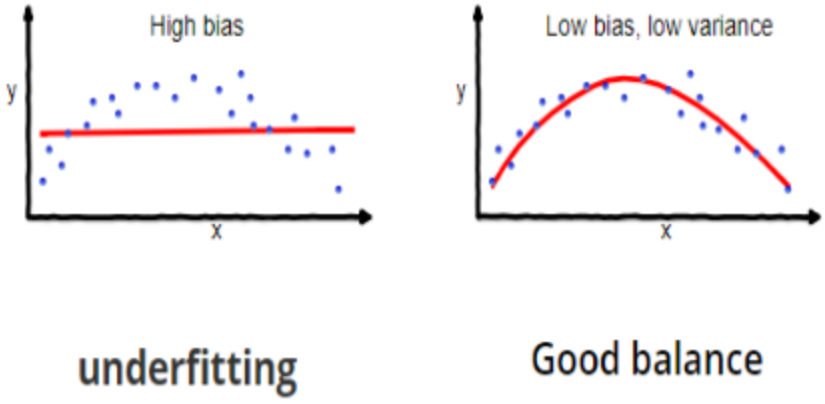
Señal y ruido



En machine learning, la señal y el ruido se refieren a la información relevante y útil en los datos de entrenamiento **(señal)** y la información irrelevante o no deseada que puede interferir con la capacidad del modelo para aprender y hacer predicciones precisas **(ruido)**.

La suposición de todos los modelos predictivos es que hay una relación entre las características (X) y el objetivo (y). Esto se llama la señal. Un modelo trata de encontrar esa señal.
Por ejemplo, si estamos entrenando un modelo de clasificación de imágenes para reconocer perros y gatos, la señal serían las características distintivas de cada animal, como la forma de las orejas, la textura del pelaje, etc. El ruido podría ser la variabilidad en las condiciones de iluminación o las posiciones de los animales en las imágenes.

Sesgo/subajuste



Identificar el subajuste

Sabrán que el modelo está subajustado cuando realiza datos de entrenamiento y de prueba deficientemente.

Por ejemplo, cuando evaluaron la puntuación R2(para modelos de regresión) del modelo en los conjuntos de entrenamiento y de prueba, encontraron que ambos estaban alrededor de 0,35. O quizás el error absoluto medio es más grande que el valor medio de la columna objetiva.

Causas del subajuste

Un modelo es demasiado simple (véase el gráfico anterior).
No hay suficientes datos.
Las características no correlacionan bien con el objetivo.

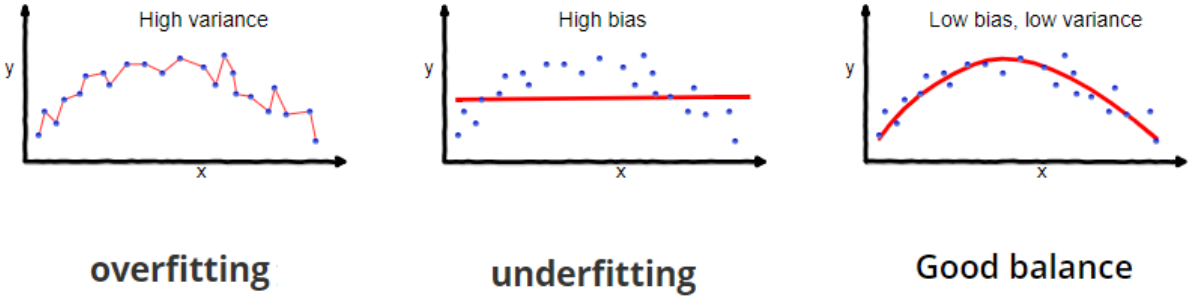
Por ejemplo, si crean un modelo de árbol de decisión con una profundidad máxima de 2 (en un modelo simple) para predecir la temperatura de mañana usando los precios de las acciones de hoy (las características no correlacionan con el objetivo) y un conjunto de entrenamiento de 5 muestras (no son datos suficientes), es probable que el modelo tenga un sesgo alto.

Reducir los sesgos

Pueden combatir el sesgo al:

- Aumentar la complejidad del modelo
- Añadir más datos
- Añadir características con una alta correlación al objetivo.
- Varianza/sobreajuste

Varianza/sobreajuste



Un modelo con una varianza alta puede hacer buenas predicciones usando algunos datos, pero no otros. Un modelo con una varianza alta tiene un sobreajuste en los datos de entrenamiento. Se ajusta al ruido que no es parte de la señal verdadera. Un modelo sobreajustado no será capaz de hacer buenas predicciones en los nuevos datos, como un conjunto de prueba. Es preciso con algunos y no con otros. Una varianza alta suele significar que pueden mejorar el rendimiento de sus modelos en el conjunto de prueba, que es lo más importante.

Sabrán que sus modelos están sobreajustados en los datos de entrenamiento y en la varianza alta si el modelo funciona mucho mejor con los datos de entrenamiento que con los de prueba.

Identificar el sobreajuste

Quizás la puntuación R2 en los datos de entrenamiento es 0,90 y la puntuación R2 en los datos de prueba es solo 0,5. O tal vez el RECM en los datos de entrenamiento es mucho más bajo que el RECM en los de prueba.

- Demasiada complejidad en un modelo (véase el gráfico anterior).
- No hay suficientes datos.

- Los datos de entrenamiento no son unas muestras representativas de la población, todos los datos nuevos que el modelo pueda encontrar.
- Observen que en el gráfico anterior el modelo se ha ajustado a cada uno de los puntos de datos. Se ha ajustado al ruido en los datos en lugar de la señal.

Combatir con el sobreajuste

- Disminuir la complejidad del modelo
- Agregar regularización (como la reducción de la dimensionalidad o la penalización L1/L2)
- Agregar más datos al conjunto de entrenamiento
- Garantizar que el conjunto de entrenamiento es un subconjunto representativo de todos los datos que el modelo encontrará.

Tipos de errores

En la biblioteca Scikit-learn (sklearn), existen varios tipos de errores que se pueden calcular utilizando la submódulo metrics. Algunos de los tipos de errores más comunes son:

- Error absoluto medio (mean_absolute_error): Este error mide la diferencia media absoluta entre las predicciones y los valores reales.
- Error cuadrático medio (mean_squared_error): Este error mide el promedio de las diferencias al cuadrado entre las predicciones y los valores reales.
- Raíz del error cuadrático medio (mean_squared_error con sqrt): Este error es la raíz cuadrada del error cuadrático medio y se utiliza a menudo porque tiene las mismas unidades que los valores reales.
- Error de mediana absoluta (median_absolute_error): Este error mide la mediana de las diferencias absolutas entre las predicciones y los valores reales.
- Precisión (accuracy_score): Este error se utiliza comúnmente para clasificación y mide la fracción de predicciones correctas.

- Recall (recall_score): Este error se utiliza comúnmente para clasificación y mide la fracción de positivos reales que se identifican correctamente.
- F1 score (f1_score): Este error se utiliza comúnmente para clasificación y combina la precisión y el recall en una sola métrica.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error,
accuracy_score, recall_score, f1_score
import numpy as np

# Valores reales
y_true = np.array([3, 2.5, 5, 7, 8.2, 6.5])

# Predicciones
y_pred = np.array([2.5, 2.8, 4.9, 6.9, 7.9, 6.6])

# Error absoluto medio
mae = mean_absolute_error(y_true, y_pred)
print("Mean Absolute Error: ", mae)  Mean Absolute Error:  0.4666666666666668

# Error cuadrático medio
mse = mean_squared_error(y_true, y_pred)
print("Mean Squared Error: ", mse) Mean Squared Error:  0.3627666666666667

# Raíz del error cuadrático medio
rmse = np.sqrt(mse)
print("Root Mean Squared Error: ", rmse) Root Mean Squared Error:  0.6021288368258856

# Error de mediana absoluta
medae = median_absolute_error(y_true, y_pred)
```

print("Median Absolute Error: ", medae) **Median Absolute Error: 0.2999999999999998**

# Precisión (clasificación binaria)
y_true_cls = np.array([1, 0, 1, 1, 0, 1])
y_pred_cls = np.array([1, 0, 0, 1, 1, 0])
acc = accuracy_score(y_true_cls, y_pred_cls)
print("Accuracy: ", acc) **Accuracy: 0.5**

# Recall (clasificación binaria)
rec = recall_score(y_true_cls, y_pred_cls)
print("Recall: ", rec) **Recall: 0.6666666666666666**

# F1 score (clasificación binaria)
f1 = f1_score(y_true_cls, y_pred_cls)
print("F1 Score: ", f1) **F1 Score: 0.39999999999999997**