INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Unidad Profesional Adolfo López Mateos

INGENIERIA EN SISTEMAS COMPUTACIONALES

Ejercicio de laboratorio 5

“Ejercicio de laboratorio 5 entrenamiento de un clasificador sencillo”

Presentan

GOMEZ HERNANDEZ ALAN JAVIER

HERNÁNDEZ PÉREZ JUAN MANUEL

JIMÉNEZ CRUZ DANIEL

RIVAS CARRERA DIANA LAURA

PROFESOR

FLORIANO GARCÍA ANDRÉS

Octubre 2023

INTRODUCCIÓN

¿Qué es machine learning?

Machine learning es una forma de la IA que permite a un sistema aprender de los datos en lugar de aprender mediante la programación explícita. Sin embargo, machine learning no es un proceso sencillo. Conforme el algoritmo ingiere datos de entrenamiento, es posible producir modelos más precisos basados en datos. Un modelo de machine learning es la salida de información que se genera cuando entrena su algoritmo de machine learning con datos. Después del entrenamiento, al proporcionar un modelo con una entrada, se le dará una salida. Por ejemplo, un algoritmo predictivo creará un modelo predictivo. A continuación, cuando proporcione el modelo predictivo con datos, recibirá un pronóstico basado en los datos que entrenaron al modelo.

Aprendizaje iterativo

Machine learning permite modelos a entrenar con conjuntos de datos antes de ser implementados. Algunos modelos de machine learning están online y son continuos. Este proceso iterativo de modelos online conduce a una mejora en los tipos de asociaciones hechas entre los elementos de datos. Debido a su complejidad y tamaño, estos patrones y asociaciones podrían haber sido fácilmente pasados por alto por la observación humana. Después de que un modelo ha sido entrenado, se puede utilizar en tiempo real para aprender de los datos. Las mejoras en la precisión son el resultado del proceso de entrenamiento y la automatización que forman parte del machine learning.

Enfoques hacia el machine learning

Las técnicas de machine learning son necesarias para mejorar la precisión de los modelos predictivos. Dependiendo de la naturaleza del problema empresarial que se está atendiendo, existen diferentes enfoques basados en el tipo y volumen de los datos. En esta sección, discutimos las categorías del machine learning.

Aprendizaje supervisado

El aprendizaje supervisado comienza típicamente con un conjunto establecido de datos y una cierta comprensión de cómo se clasifican estos datos. El aprendizaje supervisado tiene la intención de encontrar patrones en datos que se pueden aplicar a un proceso de analítica. Estos datos tienen características etiquetadas que definen el significado de los datos. Por ejemplo, se puede crear una aplicación de machine learning con base en imágenes y descripciones escritas que distinga entre millones de animales.

Aprendizaje no supervisado

El aprendizaje no supervisado se utiliza cuando el problema requiere una cantidad masiva de datos sin etiquetar. Por ejemplo, las aplicaciones de redes sociales, tales como Twitter, Instagram y Snapchat, tienen grandes cantidades de datos sin etiquetar. La comprensión del significado detrás de estos datos requiere algoritmos que clasifican los datos con base en los patrones o clústeres que encuentra. El aprendizaje no supervisado lleva a cabo un proceso iterativo, analizando los datos sin intervención humana. Se utiliza con la tecnología de detección de spam en e-mails. Existen demasiadas variables en los e-mails legítimos y de spam para que un analista etiquete una cantidad

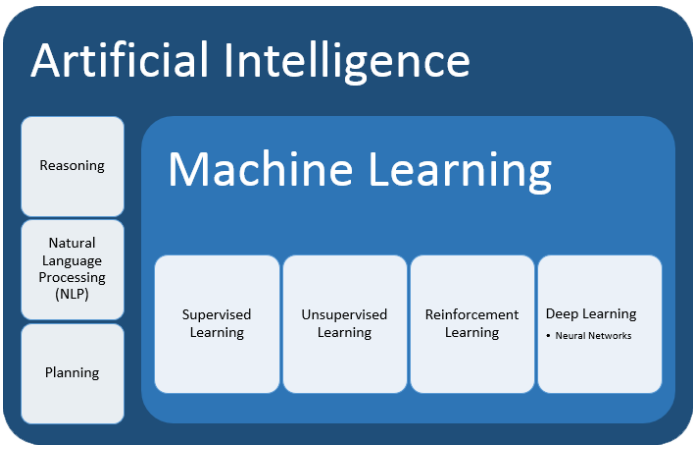
masiva de e-mail no solicitado. En su lugar, los clasificadores de machine learning, basados en clustering y asociación, se aplican para identificar e-mail no deseado.

Aprendizaje de refuerzo

El aprendizaje de refuerzo es un modelo de aprendizaje conductual. El algoritmo recibe retroalimentación del análisis de datos, conduciendo el usuario hacia el mejor resultado. El aprendizaje de refuerzo difiere de otros tipos de aprendizaje supervisado, porque el sistema no está entrenado con el conjunto de datos de ejemplo. Más bien, el sistema aprende a través de la prueba y el error. Por lo tanto, una secuencia de decisiones exitosas conduce al fortalecimiento del proceso, porque es el que resuelve el problema de manera más efectiva.

Deep learning

El deep learning es un método específico de machine learning que incorpora las redes neuronales en capas sucesivas para aprender de los datos de manera iterativa. El deep learning es especialmente útil cuando se trata de aprender patrones de datos no estructurados. Las redes neuronales complejas de deep learning están diseñadas para emular cómo funciona el cerebro humano, así que las computadoras pueden ser entrenadas para lidiar con abstracciones y problemas mal definidos. Las redes neuronales y el deep learning se utilizan a menudo en el reconocimiento de imágenes, voz y aplicaciones de visión de computadora.



DESARROLLO

1. Empleando el mismo enfoque revisado en la clase de ayer, realiza el entrenamiento de un clasificador sencillo con los datos del archivo train.csv
    a. Prueba primero con la variable petallength,
    b. Luego prueba con petalwidth.
2. Una vez que definiste el umbral de separación, prueba con los datos del archivo test.csv.
    a. Clasifica los datos de la variable petallength, compara con la clase real y cuenta cuántas patrones clasificaste correctamente.
    b. Clasifica los datos de la variable petalwidth, compara con la clase real y cuenta cuántos patrones clasificaste correctamente

Justificación.

Optamos por usar Naive Bayes como algoritmo dado que es una técnica de clasificación simple pero poderosa en el campo del aprendizaje automático. Para este caso es importante mencionar las ventajas y las desventajas.

Ventajas	Desventajas
• Facilidad de implementación y comprensión: Naive Bayes es un algoritmo relativamente simple de entender e implementar. Esto lo hace adecuado para problemas de clasificación tanto para principiantes como para expertos. • Eficiencia en tiempo de ejecución: Naive Bayes es rápido y eficiente en términos de tiempo de ejecución, lo que lo hace útil para conjuntos de datos grandes. • Requiere menos datos de entrenamiento: En comparación con algunos algoritmos más complejos, Naive Bayes puede funcionar bien con conjuntos de datos más pequeños. Esto es especialmente útil en situaciones en las que hay limitaciones en la cantidad de datos disponibles. • Manejo de características categóricas y numéricas: Naive Bayes puede manejar tanto características categóricas como numéricas. Esto lo hace versátil en una variedad de aplicaciones. • Puede manejar atributos irrelevantes: El algoritmo puede manejar atributos irrelevantes o no informativos, ya que asume independencia entre las características.	• Sesgo de independencia: La principal limitación del algoritmo Naive Bayes es la suposición de independencia condicional entre las características (de ahí el "naive" en su nombre). En la realidad, las características rara vez son completamente independientes, lo que puede llevar a un rendimiento subóptimo en casos en los que las dependencias son significativas. • No modela relaciones complejas: Naive Bayes es un modelo probabilístico simple y no puede capturar relaciones complejas entre características. Por lo tanto, puede no ser adecuado para problemas en los que las relaciones entre las características son críticas. • Sensible a datos desequilibrados: Naive Bayes puede verse afectado por datos desequilibrados, donde una clase tiene muchas más instancias que otras. Esto puede resultar en una clasificación sesgada hacia la clase dominante. • Necesidad de manejar valores faltantes: El algoritmo no maneja naturalmente valores faltantes en los datos. Deben abordarse antes de aplicar Naive Bayes.

Naive Bayes es una excelente elección para problemas de clasificación simples o en situaciones en las que se dispone de un conjunto de datos limitado. Sin embargo, en problemas más complejos con dependencias significativas entre características, otros algoritmos de clasificación más avanzados, como Support Vector Machines (SVM), Random Forests o Redes Neuronales, pueden superar el rendimiento de Naive Bayes.

Train.csv

petallength	petalwidth	class
1.4	0.2	Iris-setosa
1.4	0.2	Iris-setosa
1.3	0.2	Iris-setosa
1.5	0.2	Iris-setosa
1.4	0.2	Iris-setosa
1.7	0.4	Iris-setosa
1.4	0.3	Iris-setosa
1.5	0.2	Iris-setosa
1.4	0.2	Iris-setosa
1.5	0.1	Iris-setosa
1.5	0.2	Iris-setosa
1.6	0.2	Iris-setosa
1.4	0.1	Iris-setosa
1.4	0.2	Iris-setosa
4.7	1.4	Iris-versicolor
4.5	1.5	Iris-versicolor
4.9	1.5	Iris-versicolor
4	1.3	Iris-versicolor
4.6	1.5	Iris-versicolor
4.5	1.3	Iris-versicolor
4.7	1.6	Iris-versicolor
3.3	1	Iris-versicolor
4.6	1.3	Iris-versicolor
3.9	1.4	Iris-versicolor
3.5	1	Iris-versicolor
4.2	1.5	Iris-versicolor
4	1	Iris-versicolor
4.7	1.4	Iris-versicolor
6	2.5	Iris-virginica
5.1	1.9	Iris-virginica
5.9	2.1	Iris-virginica
5.6	1.8	Iris-virginica
5.8	2.2	Iris-virginica
6.6	2.1	Iris-virginica
4.5	1.7	Iris-virginica
6.3	1.8	Iris-virginica
5.8	1.8	Iris-virginica
6.1	2.5	Iris-virginica
5.1	2	Iris-virginica
5.3	1.9	Iris-virginica
5.5	2.1	Iris-virginica
5	2	Iris-virginica

Test.csv

petallength	petalwidth	class
1.1	0.1	Iris-setosa
1.2	0.2	Iris-setosa
1.5	0.4	Iris-setosa
1.3	0.4	Iris-setosa
1.4	0.3	Iris-setosa
1.7	0.3	Iris-setosa
3.6	1.3	Iris-versicolor
4.4	1.4	Iris-versicolor
4.5	1.5	Iris-versicolor
4.1	1	Iris-versicolor
4.5	1.5	Iris-versicolor
3.9	1.1	Iris-versicolor
5.1	2.4	Iris-virginica
5.3	2.3	Iris-virginica
5.5	1.8	Iris-virginica
6.7	2.2	Iris-virginica
6.9	2.3	Iris-virginica
5	1.5	Iris-virginica

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

import joblib
```

Importaremos las librerías para el desarrollo de la practica;

- **pandas** se utiliza para la manipulación de datos y carga de datos tabulares.
- **scikit-learn** se utiliza para crear, entrenar y evaluar modelos de aprendizaje automático, específicamente el modelo de Naive Bayes.
- **joblib** se utiliza para guardar el modelo entrenado en un archivo para su posterior uso.

scikit-learn (también conocido como sklearn) es una biblioteca de Python ampliamente utilizada para machine learning y aprendizaje automático. GaussianNB es una clase dentro de scikit-learn que implementa el algoritmo de Naive Bayes gaussiano, que es una técnica de clasificación. Se importa para crear y entrenar un modelo de Naive Bayes gaussiano en tu código. ***Naive Bayes*** es útil para tareas de clasificación.

classification_report es una función proporcionada por **scikit-learn** que se utiliza para generar un informe detallado de las métricas de rendimiento de un modelo de clasificación. Este informe incluye información sobre precisión, recall, F1-score y otras métricas que ayudan a evaluar qué tan bien el modelo clasifica los datos. Se importa para evaluar el rendimiento del modelo Naive Bayes.

joblib es una biblioteca de Python que se utiliza para guardar y cargar objetos de Python en el disco. En tu código, está importado con el propósito de guardar el modelo entrenado en un archivo con la extensión ".pkl". Esto es útil para conservar el modelo para su uso posterior sin tener que volver a entrenarlo desde cero cada vez que se necesite.

```
# Cargar los datos de entrenamiento desde un archivo Excel
train_data = pd.read_csv('./train.csv')  # Reemplaza 'train.xlsx' con
el nombre de tu archivo Excel de entrenamiento

# Cargar los datos de prueba desde un archivo Excel
test_data = pd.read_csv('./test.csv')  # Reemplaza 'test.xlsx' con el
nombre de tu archivo Excel de prueba
```

En esta parte del código carga los datos de entrenamiento desde un archivo Excel train_data = pd.read_csv('./train.csv'): Esta línea utiliza la función read_csv de la biblioteca Pandas para cargar datos desde un archivo CSV. Después carga los datos de prueba desde un archivo Excel

```
# Separar las características (X) y la etiqueta (y) para los datos de
entrenamiento
X_train = train_data.drop('class', axis=1)
y_train = train_data['class']

# Separar las características (X) y la etiqueta (y) para los datos de
prueba
X_test = test_data.drop('class', axis=1)
y_test = test_data['class']
```

En esta parte separa las características y la etiqueta para los datos de entrenamiento. **X_train = X_train y y_train contienen las características y las etiquetas de entrenamiento, respectivamente.**

X_test y y_test contienen las características y las etiquetas de prueba, respectivamente.

Esto prepara los datos para entrenar y evaluar un modelo de aprendizaje automático, en este caso, un modelo de Naive Bayes gaussiano, que se entrena con X_train y y_train y se evalúa con X_test y y_test.

```
# Crear y entrenar el modelo Naive Bayes
model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

# Evaluar el modelo (opcional)
accuracy = model.score(X_test, y_test)
report = classification_report(y_test, y_pred)

print("Precisión del modelo:", accuracy)
print(f'Exactitud del modelo: {accuracy}')
print("Informe de precisión:", report)
```

En esta parte de código crea un modelo Naive Bayes gaussiano, lo entrena con datos de entrenamiento, utiliza el modelo para predecir etiquetas en datos de prueba y opcionalmente evalúa el rendimiento del modelo utilizando métricas de precisión y un informe de clasificación. La evaluación es útil para comprender qué tan bien el modelo está realizando la tarea de clasificación en los datos de prueba.

Código Completo

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

import joblib

# Cargar los datos de entrenamiento desde un archivo Excel
train_data = pd.read_csv('./train.csv')  # Reemplaza 'train.xlsx' con
el nombre de tu archivo Excel de entrenamiento

# Cargar los datos de prueba desde un archivo Excel
test_data = pd.read_csv('./test.csv')  # Reemplaza 'test.xlsx' con el
nombre de tu archivo Excel de prueba

# Separar las características (X) y la etiqueta (y) para los datos de
entrenamiento
X_train = train_data.drop('class', axis=1)
y_train = train_data['class']

```

```
# Separar las características (X) y la etiqueta (y) para los datos de
prueba
X_test = test_data.drop('class', axis=1)
y_test = test_data['class']

# Crear y entrenar el modelo Naive Bayes
model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

# Evaluar el modelo (opcional)
accuracy = model.score(X_test, y_test)
report = classification_report(y_test, y_pred)

print("Precisión del modelo:", accuracy)
print(f'Exactitud del modelo: {accuracy}')
print("Informe de precisión:", report)

# Guardar el modelo en un archivo .pkl
#joblib.dump(model, 'modelo_naive_bayes.pkl')
```

PRUEBAS DE FUNCIONAMIENTO

```
Precisión del modelo: 0.9444444444444444
Exactitud del modelo: 0.9444444444444444
Informe de precisión:                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         6
Iris-versicolor       0.86      1.00      0.92         6
 Iris-virginica       1.00      0.83      0.91         6

       accuracy                           0.94        18
      macro avg       0.95      0.94      0.94        18
   weighted avg       0.95      0.94      0.94        18
```

Podemos ver que la precisión del modelo es aproximadamente 0.9444, lo que significa que el modelo clasificó correctamente alrededor del 94.44% de las muestras en el conjunto de prueba.

En este caso, la exactitud es la misma que la precisión (0.9444). Estas dos métricas son equivalentes en el contexto de una clasificación binaria o de múltiples clases.

Y tenemos el informe de precisión.

Dado que estamos trabajando con un problema de clasificación de tres clases, donde las clases son "Iris-setosa", "Iris-versicolor" e "Iris-virginica".

- **precision**: La precisión mide la proporción de predicciones positivas correctas en relación con todas las predicciones positivas. Como el valor es de 1.00 significa que el modelo no cometió errores al predecir esta clase.
- **recall**: El recall (también conocido como sensibilidad) mide la proporción de instancias positivas que fueron correctamente clasificadas en relación con todas las instancias reales positivas. Nos dio un valor de 1.00 significa que el modelo identificó todas las instancias reales positivas.
- **f1-score**: El puntaje F1 es una medida que combina precisión y recall en un solo valor. Es útil cuando se busca un equilibrio entre precisión y recall.
- **support**: La cantidad de muestras en el conjunto de prueba que pertenecen a esa clase.

Accuracy (exactitud): La exactitud general del modelo en todo el conjunto de prueba es del 0.94. Esto significa que el 94% de las muestras en el conjunto de prueba se clasificaron correctamente.

Promedio ponderado (weighted avg): En este caso, el promedio ponderado de la precisión, recall y f1-score es del 0.95.

Promedio macro (macro avg): En este caso, el promedio macro de la precisión, recall y f1-score es del 0.94.

CONCLUSIÓN

Para esta práctica tuvimos una introducción sobre machine learning donde los principales conceptos están entorno a los tipos de aprendizaje supervisado, no supervisado y semi supervisado. Dado que exploramos una solución para el aprendizaje supervisado se entreno el clasificador utilizando datos de entrenamiento del archivo "train.csv". Se realizaron dos pruebas separadas, una utilizando la variable "petallength" y la otra utilizando la variable "petalwidth". Durante esta etapa, se determinaron los umbrales de separación óptimos para cada variable. Al igual que se propone el uso de un clasificador Naive Bayes gaussiano para realizar tareas de clasificación en un conjunto de datos de flores Iris. Ahora con esto el modelo entrenado se utilizó para clasificar los datos del archivo "test.csv" en función de las variables "petallength" y "petalwidth". Podemos decir que se ha demostrado la importancia de seleccionar características adecuadas para lograr un rendimiento óptimo en problemas de clasificación. La práctica destaca la importancia de adaptar las estrategias de clasificación según las características y requisitos del problema en cuestión.

REFERENCIAS

"Machine Learning". IBM in Deutschland, Österreich und der Schweiz | IBM. Accedido el 18 de octubre de 2023. [En línea]. Disponible: https://www.ibm.com/mx-es/analytics/machine-learning

"¿Qué es el aprendizaje supervisado? | IBM". IBM in Deutschland, Österreich und der Schweiz | IBM. Accedido el 18 de octubre de 2023. [En línea]. Disponible: https://www.ibm.com/mx-es/topics/supervised-learning