1. ML

September 2, 2025

1 ML

1.0.1 El aprendizaje automático (machine learning ML) es un método basado en algoritmos para analizar datos para buscar patrones y hacer predicciones precisas.

1.0.2 Los algoritmos de ML son entrenados de diferentes maneras y se pueden dividen en tres categorías.

1. **No supervisados**
2. **Supervisados**
3. **Reforzamiento**

1.1 1. Algoritmos de ML no supervisados

ML no supervisado se basa en la idea de que una máquina puede aprender sin ninguna guía. Para el aprendizaje, utiliza datos que no están estructurados ni procesados. La limitación de estos algoritmos es que no tienen un punto de partida y por lo tanto, solo pueden realizar un número reducido de tareas. Las dos principales tareas que pueden realizar son:

1.1.1 a. Agrupamiento (clustering)

Este algoritmo puede diferenciar dos cosas en función de sus características y separarlas en grupos (clusters). La agrupación en clústeres es excelente para resolver tareas como: filtrado de correo, detección de fraudes, agrupación jerárquica en clústeres para el análisis de documentos, etc.

1.1.2 a. Reducción de dimensionalidad

Este tipo de algoritmo de procesamiento y la simplificación de los datos al disminuir el número de características. El modelo de reducción de dimensionalidad reduce las características que no son esenciales para la tarea en cuestión, pero deja intactas la estructura y las características principales de los datos.

La reducción de ruido y la visualización de datos son tareas comunes para los algoritmos de reducción de dimensionalidad. También se usa comúnmente como un paso intermedio en proyectos de ML más complejos.

1.2 2. Algoritmos de aprendizaje automático supervisados

A diferencia del aprendizaje no supervisado, los algoritmos supervisados requieren datos etiquetados, lo que le indicará al algoritmo cual es su significado. Esto significa que los modelos se entrenan en función de los datos que han sido procesados.

El procesamiento de los datos es la supervisión que se realiza sobre el proceso de entrenamiento, de ahí el nombre de aprendizaje supervisado. Hay bastantes tipos de algoritmos los mas utilizados son: los de regresión, clasificación y pronóstico (forecasting).

1.2.1 a. Regresión

Se requiera el análisis de valores continuos para encontrar una correlación entre diferentes variables. La regresión ayuda a buscar esta correlación y predecir una salida.

Este tipo de algoritmo supervisado se usa comúnmente para predecir los precios o el valor de ciertos objetos en función de un conjunto de sus características.

1.2.2 b. Clasificación

Similar a la agrupación no supervizada, la clasificación permite entrenar al algoritmo para agrupar diferentes objetos (valores) en categorías (o clases). La diferencia es que, ahora, el algoritmo sabe qué clase contiene qué objetos.

A diferencia de la regresión, la clasificación se basa en un número limitado de valores. Puede ser binario o multiclase .

1.2.3 c. Pronóstico

Los algoritmos de pronóstico pueden "predecir el futuro", ya que pueden analizar los datos "pasados y presentes" en profundidad, buscar patrones ocultos y hacer predicciones basadas en este análisis.

El análisis de tendencias es el fuerte de este tipo de algoritmo de aprendizaje automático. Es por eso que el pronóstico se usa comúnmente en negocios y finanzas.

1.3 3. Algoritmos de aprendizaje automático semisupervisados

Hay casos en los que la combinación de los dos tipos de algoritmos puede brindarl mejores resultados, esto se debe a las características principales de cada tipo de algoritmo: el aprendizaje no supervisado aporta *simplicidad y eficiencia*, mientras que el aprendizaje supervisado tiene que ver con la *flexibilidad y los objetivos integrales*.

Cuando se combinan dos tipos de diferentes algoritmos, se obtiene un aprendizaje **semisupervisado**. Este tipo de algoritmo ML le permite reducir significativamente el costo financiero, humano y de tiempo para anotar los datos. Al mismo tiempo, los algoritmos de aprendizaje semisupervisados no están tan restringidos en la elección de tareas como los algoritmos de aprendizaje supervisado.

2 El mejor algoritmo de aprendizaje automático

1. Comprenda el objetivo de su proyecto

Considerar el tipo de proyecto con el que está tratando: ¿qué tipo de salida necesita? ¿Necesitas un algoritmo de predicción basado en los datos anteriores? Recurra a los algoritmos de pronóstico supervisados. ¿Está buscando un modelo de reconocimiento de imágenes que funcione con fotos de baja calidad? La reducción de la dimensionalidad en combinación con la clasificación le ayudará con ello. ¿Necesitas enseñarle a tu modelo a jugar un juego nuevo? Un algoritmo de refuerzo será su mejor apuesta.

2. Analizar los datos por tamaño, procesamiento y anotación requerida

Qué entradas se tienen?. ¿Cómo son tus datos? ¿Es crudo, simplemente recolectado de donde sea y requiere procesamiento? ¿Es parcial, sucio y desestructurado? ¿O ya tiene un gran conjunto de datos anotados en sus manos? ¿Tiene suficientes datos o se requiere una recopilación adicional (o incluso una recopilación desde cero)? ¿Necesita dedicar tiempo a preparar sus datos para el proceso de capacitación o está listo para comenzar?. Los datos insuficientes, de baja calidad y sin procesar generalmente no se prestan a un gran entrenamiento de un algoritmo supervisado. Debe decidir si desea dedicar tiempo y recursos a preparar los mejores datos que pueda antes de comenzar el proceso de capacitación. De lo contrario, puede optar por algoritmos no supervisados, pero tenga en cuenta las limitaciones de dicha elección.

3. Evalúar el Tiempo de Entrenamiento

¿Lo necesita rápido incluso si eso significa una menor calidad de entrenamiento (y, respectivamente, predicciones)? Más datos y de mayor calidad conducen a una mejor capacitación. ¿Se puede asignar el tiempo necesario para la formación adecuada?

4. Saber la linealidad de los datos

Los algoritmos lineales (como la regresión lineal o las máquinas de vectores de soporte) son más simples y rápidos de entrenar. Sin embargo, no suelen usarse para problemas más complejos ya que tratan con datos lineales. Si los datos son multifacéticos, multidimensionales y tienen muchas correlaciones que se cruzan, es posible que los algoritmos lineales no sean suficientes para su tarea.

5. Decida la cantidad de funciones y parámetros

¿Qué tan complejo y preciso debe ser el modelo al final? ¿Se pueden especificar más características y parámetros para que el modelo los interprete y se entrene por más tiempo?. Darle a su algoritmo más tiempo para aprender puede ser una buena inversión en su futura precisión e interpretabilidad de salida.

3 Pasos para construir y usar un modelo

1. Pre-procesameinto de los datos

```
a. Importar los datos
b. Limpiar los datos
c. Separar en datos de entrenamiento y de prueba
d. Escalamiento de las caracteristicas
```

2. Modelado

```
a. Construir el modelo
b. Entrenar el modelo
c. Hacer prediciones con el modelo
```

3. Evaluación

```
a. Calcular las métricas de rendimiento del modelo
b. Concluir si el modelo es eficiente o no
```

4 1. Importar bibliotecas

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

5 2. Importat datos

https://www.kaggle.com/datasets/rakeshrau/social-network-ads

Datos categóricos para determinar si un usuario compró un producto

[2]:
```
dataset= pd.read_csv('Social_Network_Ads.csv')

X = dataset.iloc[:, :-1].values # caracteristicas
y = dataset.iloc[:, -1].values # variable a predecir
dataset
```

[2]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
..	…	…	…
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

[400 rows x 3 columns]

6 3. Separar datos en datos de entrenamiento y datos de prueba

[3]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
↪random_state = 0)
```

7 4. Escalamiento de las características

[4]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)
```

8 5. Entrenar el modelo

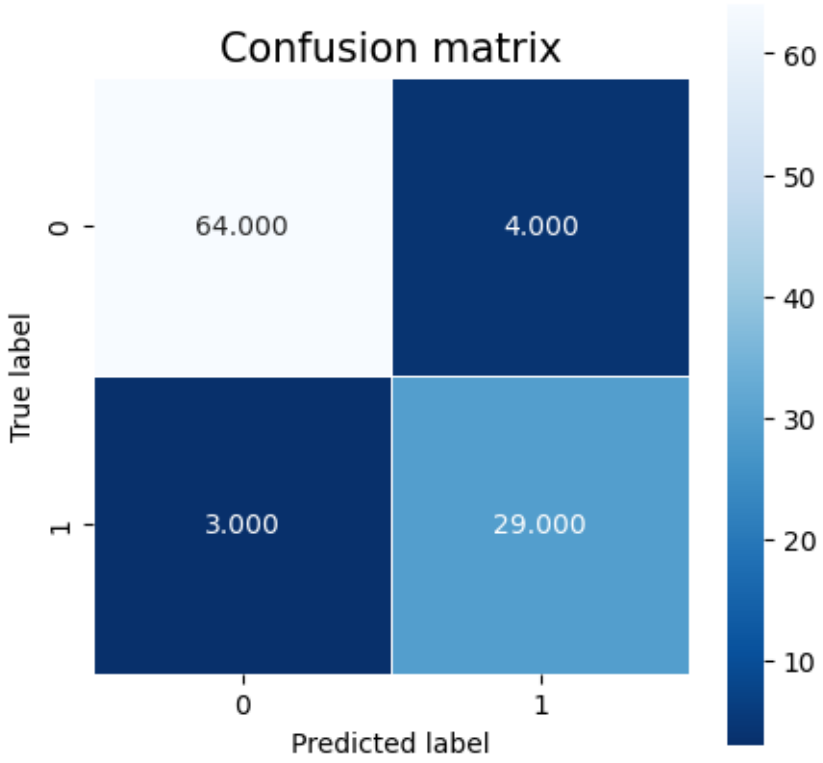
[5]:
```
from sklearn.svm import SVC # maquinas de vector soporte

classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

[5]:
```
SVC(random_state=0)
```

9 6. Obtener la matriz de confusión

[6]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap =␣
 ↪'Blues_r')
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.title('Confusion matrix', size = 15)
y_pred = classifier.predict(X_test)
```



10 7. Validación del entrenamiento del modelo

[7]:
```
from sklearn.model_selection import cross_val_score

accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train,␣
 ↪cv = 10,n_jobs = -1)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

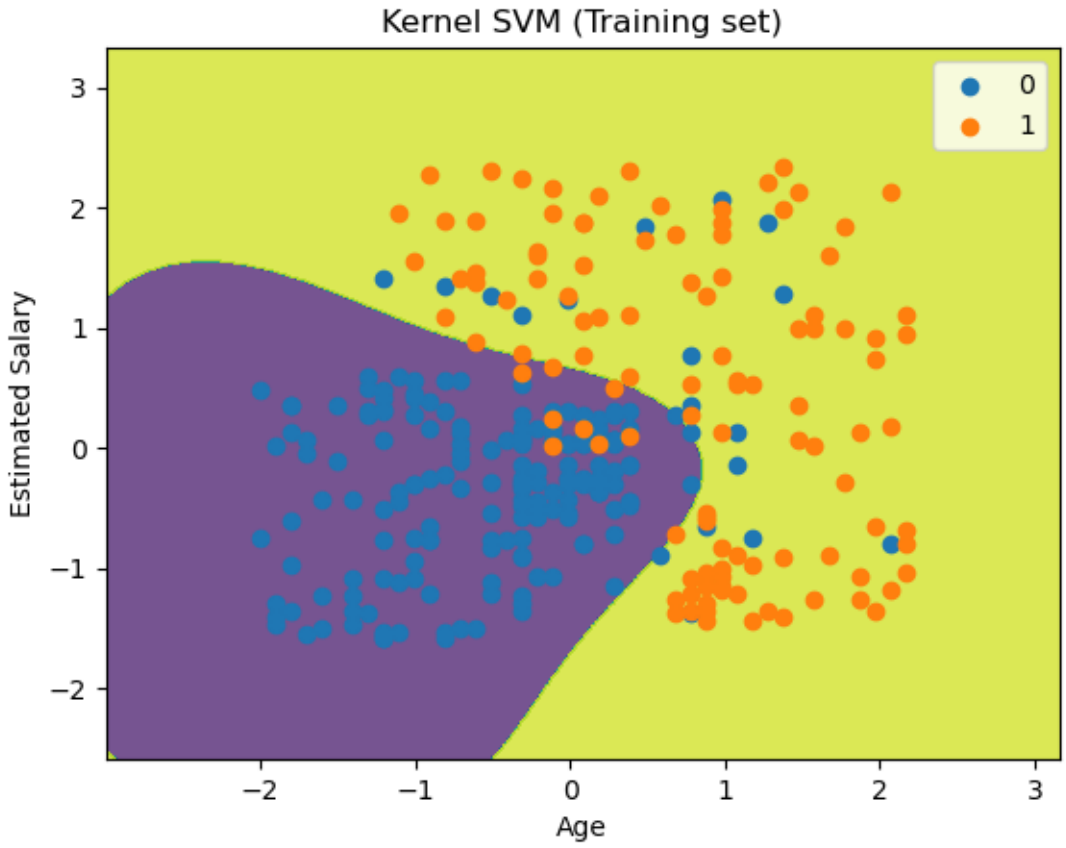
```
Accuracy: 90.33 %
Standard Deviation: 6.57 %
```

11 8. Visualizacion del resultado del entrenamiento

[8]:
```
from matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,␣
 ↪0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,␣
 ↪1].max() + 1, step = 0.01))
```

```
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).
 ↪reshape(X1.shape),
             alpha = 0.75)
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                label = j)
plt.title('Kernel SVM (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```



12 9. Visualización del resultado de prueba

[9]:
```
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:,␣
 ↪0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:,␣
 ↪1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).
 ↪reshape(X1.shape),
             alpha = 0.75)
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                label = j)
plt.title('Kernel SVM (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```