Guía de estudio en Machine Learning y Deep Learning

Carlos E Martínez-Rodríguez

14 de noviembre de 2023

Índice

1. Análisis de Componentes Principales

1.1. Introduction to Principal Components Analysis - Kristin L. Sainani

Principal components analysis (PCA) is a powerful statistical tool that can help researchers analyze datasets with many highly related predictors. PCA is a data reduction technique -that is, it reduces a larger set of predictor variables to a smaller set with minimal loss of information. PCA may be applied before running regression analyses or for exploratory purposes to help researchers understand relationships among their variables or discover patterns in their data. Besides compressing the data, PCA analysis also reveals clusters of variables that are highly related, which can give investigators a deeper understanding of their data

Nota 1 *To illustrate the value of PCA, I will apply the technique to some data pertaining to 91 female adolescent runners. These data come from a larger real dataset [1] but have been modified for use in classroom examples, and thus the results presented here are only for teaching purposes. The researchers measured a large number of potential predictors of bone density and stress fractures, including variables relating to body size and composition, training, per- formance, menstrual function, diet, and eating behaviors. An examination of the correlation coefficients between these 11 variables shows that many are highly related, even across different groups of variables. The application of PCA to these data before performing regression analyses has several benefits. The researchers have collected multiple measurements that reflect the same construct—for example, running performance or menstrual function. If they enter all of these variables into a regression model, they increase the risks of overfitting and type I errors (chance findings). However, if they ignore or discard variables, they lose valuable infor- mation. PCA analysis instead compresses the 11 original variables into a smaller subset of composite variables (called principal components) that capture most of the information in the original data. These new variables may then be used for further analyses. They have the added benefit of being uncorrelated with one another, which makes them easier to interpret and analyze. Before performing PCA analysis on the example data, I first imputed missing values, because observations that have any missing data will otherwise be omitted. I also converted the variables into standard deviation units (Z scores), because all the variables need to have the same units before PCA is applied.*

1.2. Principal component analysis - Herve Abdi and Lynne J. Williams

PCA analyzes a data table representing observations described by several dependent vari- ables, which are, in general, inter-correlated. Its goal is to extract the important information from the data table and to express this information as a set of new orthogonal variables called principal components. PCA also represents the pattern of similarity of the observations and the variables by displaying them as points in maps

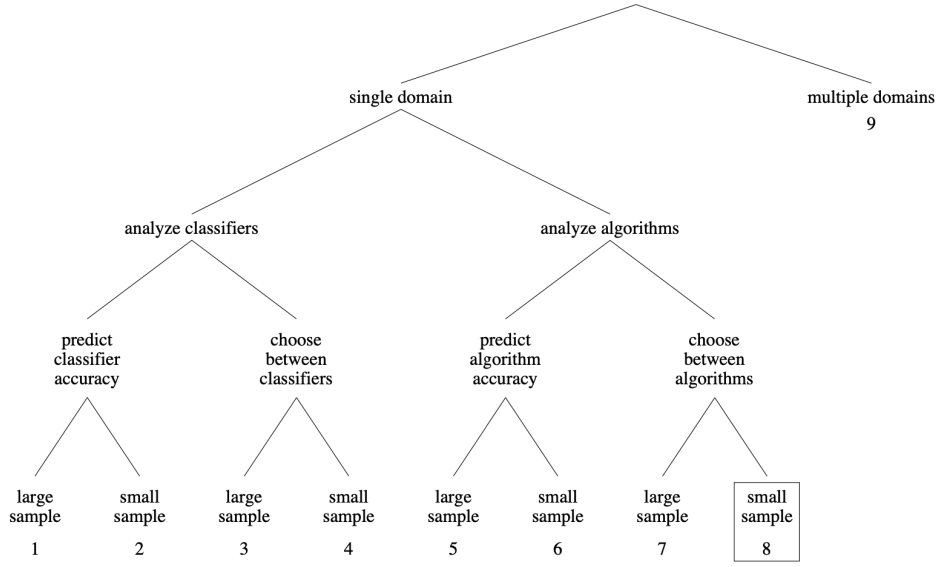
1.3. Approximate Statistical Test for comparing Supervised Classification Learning Algortihms - Dietterich Thomas

El objetivo de esta investigación es encontrar el mejor clasificador o el mejor algoritmo de aprendizaje para ser aplicado en ese dominio. Un objetivo principal en ML es encontrar algoritmos que trabajen bien en un amplio rango de aplicaciones.

Nota 2 *Un clasificador es una función que dada un ejemplo de entrada, a ese ejemplo se le asigne una de las K clases.*

Nota 3 *Un algoritmo de aprendizaje es una función que dado un conjunto de ejemplos y sus clases, construya un clasificador.*

Nota 4 *En un principio, el objetivo es encontrar el mejor clasificador y estimar su precisión en futuros ejemplos. En algunas aplicaciones se seleccionará el mejor algoritmo de aprendizaje más que el mejor clasificador.*



El siguiente nivel distingue entre estimar la precisión y elegir clasificadores o algoritmos. El nivel más bajo está relacionado con la cantidad de datos disponibles, si se tiene una gran cantidad de datos, entonces se pueden fijar un conjunto de ellos para ser considerados como datos de prueba para evaluar los clasificadores. En muchos de los casos, la cantidad de datos es limitada y se requeire utilizar todos como datos de entrada a los algoritmos de aprendizaje, esto quiere decir que se requiere utilizar algunas técnicas de remuestreo para realizar análisis estadísticos.

Nota 5 ***Nueve Preguntas***
Se supone que todos los puntos de datos se obtienen de manrea independiente de una distribución de probabilidad fija definida por el problema en particular.

Pregunta 1.1 *Supóngase que se tiene una muestra de datos grande y un clasificador C. El clasificador pudo haber sido construido utilizando parte de los datos, pero aún existen datos para conformar un conjunto de datos de prueba. Por tanto se puede medir la precisión de C en el conjunto de datos de prueba y construir un intervalo de confianza binomial. El clasificador pudo haber sido generado por cualquier método y no necesariamente por un algoritmo de aprendizaje.*

Pregunta 1.2 *Dado un conjunto de datos pequeño S, supóngase que se aplica un algoritmo de aprendizaje A a S para construir un clasificador C_A, qué tan preciso es el clasificador C_A para nuevos ejemplos. Dado que no se tienen conjuntos de datos separados no es posible responder la pregunta directamente. Es posible predecir la precisión del algoritmo A cuando se entrena en conjuntos de datos seleccionados aleatoriamente de aproximadamente el mismo tamaño que S. Si esto es posible entonces se puede predecir la precisión de C_A que fue obtenido despues de ser entrenado en S.*

Pregunta 1.3 *Dados dos clasificadores C_A y C_B y suficientes datos para tener un conjunto de datos de prueba, determinar cual clasificador será más preciso en nuevos ejemplos de prueba. Esto se puede responder midiendo la precisión de cada clasificador en los datos de prueba y aplicando la prueba de McNemar.*

Pregunta 1.4

Pregunta 1.5

Pregunta 1.6

Pregunta 1.7

Pregunta 1.8

Pregunta 1.9

2. Guia de estudio de Machine Learning

- Algoritmos de aprendizaje supervisado:
  - Regresión lineal y logística
  - Máquinas de soporte vectorial (SVM)
  - Árboles de decisión y bosques aleatorios
  - Redes neuronales
- Aprendizaje no supervisado:
  - K-Means y clustering jerárquico
  - Análisis de componentes principales (PCA)
  - Algoritmos de asociación
  - Mapas autoorganizados (SOM)
- Evaluación de modelos y métricas:
  - Precisión, sensibilidad, especificidad
  - Curvas ROC y área bajo la curva (AUC-ROC)
  - Matriz de confusión
  - Validación cruzada
- Preprocesamiento de datos:
  - Normalización y estandarización
  - Manejo de datos faltantes
  - Ingeniería de características
  - Selección de características
- Optimización de modelos:
  - Hiperparámetros y búsqueda en cuadrícula
  - Optimización bayesiana
  - Regularización
  - Redes neuronales convolucionales (CNN) y recurrentes (RNN)
- Aprendizaje por refuerzo:
  - Q-Learning
  - Algoritmos de políticas
  - Exploración y explotación
  - Funciones de valor
- Ética en el machine learning:

- Sesgo y equidad
- Transparencia y explicabilidad
- Privacidad y seguridad
- Responsabilidad en el despliegue de modelos

2.1. SVM

1. **Hiperplano:** En un espacio de características n-dimensional, un hiperplano es un subespacio de dimensión $n-1$. Para un problema de clasificación binaria, un hiperplano divide el espacio en dos regiones, asignando puntos a una clase u otra.

2. **Margen:** El margen es la distancia perpendicular desde el hiperplano a los puntos más cercanos de cada clase. SVM busca el hiperplano que maximiza este margen, lo que se traduce en una mayor robustez y generalización del modelo.

3. **Vectores de Soporte:** Estos son los puntos de datos más cercanos al hiperplano y tienen un papel crucial en la definición del margen. Cambiar estos vectores de soporte afecta directamente al modelo, y son los únicos puntos que importan para la determinación del hiperplano.

4. **Función de Decisión:** La función de decisión de SVM es el hiperplano que se utiliza para clasificar nuevos puntos de datos. Dada una entrada, la función de decisión evalúa de qué lado del hiperplano cae el punto y asigna la etiqueta correspondiente.

5. **Kernel Trick:** SVM puede manejar eficientemente datos no lineales mediante el uso de funciones de kernel. Estas funciones transforman el espacio de características original en uno de mayor dimensión, permitiendo así que los datos sean separados de manera no lineal en el espacio transformado.

6. **Parámetros de SVM:**

   - **C (Parámetro de Regularización):** Controla el equilibrio entre tener un margen más amplio y clasificar correctamente los puntos de entrenamiento.
   - **Kernel:** Define la función de kernel utilizada (lineal, polinómica, radial, etc.).
   - **Gamma (para kernels no lineales):** Controla el alcance de influencia de un solo punto de datos en la decisión.

7. **Proceso de Entrenamiento:** Dado un conjunto de datos de entrenamiento etiquetado, SVM busca el hiperplano óptimo que maximiza el margen entre las clases. Esto se realiza a través de técnicas de optimización cuadrática.

8. **SVM para Regresión:** Además de la clasificación, SVM se puede utilizar para problemas de regresión. En este caso, el objetivo es ajustar un hiperplano de modo que contenga la mayor cantidad posible de puntos dentro de un margen predefinido.

9. **Ventajas y Desventajas:**

   - **Ventajas:** Efectivo en espacios de alta dimensión, eficaz en conjuntos de datos pequeños y versátil gracias al kernel trick.

- **Desventajas:** Sensible a la escala de las características, puede ser computacionalmente costoso para grandes conjuntos de datos y requiere la elección cuidadosa de parámetros.

10. **Aplicaciones:** SVM se utiliza en una variedad de campos, como reconocimiento de escritura, clasificación de imágenes, diagnóstico médico, entre otros.

2.1.1. Elementos Matemáticos de las SVM

1. **Hiperplano:** Un hiperplano se define como $w \cdot x - b = 0$, donde w es el vector de pesos, x es el vector de entrada, y b es el sesgo.

2. **Margen:** El margen M entre un hiperplano y un punto x_i se define como $M = \frac{1}{\|w\|}|w \cdot x_i - b|$.

3. **Vectores de Soporte:** Los vectores de soporte son los puntos x_i que cumplen la condición $|w \cdot x_i - b| = 1/\|w\|$.

4. **Función de Decisión:** La función de decisión es $f(x) = w \cdot x - b$. Si $f(x) > 0$, el punto x se clasifica como clase 1; si $f(x) < 0$, se clasifica como clase -1.

5. **Kernel Trick:** La función de kernel $K(x, x')$ representa el producto escalar en un espacio de características de mayor dimensión. Ejemplos comunes incluyen el kernel lineal ($K(x, x') = x \cdot x'$), kernel polinómico ($K(x, x') = (x \cdot x' + 1)^d$), y kernel radial ($K(x, x') = \exp(-\gamma\|x - x'\|^2)$).

6. **Parámetros de SVM:**

   - C (Parámetro de Regularización): Se introduce en la función de pérdida para controlar el equilibrio entre tener un margen más amplio y clasificar correctamente los puntos de entrenamiento.
   - w (Vector de Pesos): Aprende durante el entrenamiento y define la orientación del hiperplano.
   - b (Sesgo): Parámetro de ajuste del hiperplano.
   - γ (para kernels no lineales): Controla el alcance de influencia de un solo punto de datos en la decisión.

7. **Proceso de Entrenamiento:** El proceso de entrenamiento implica la minimización de la función de pérdida, que incluye el término de regularización $C\|w\|^2$ y la función de pérdida hinge.

8. **SVM para Regresión:** Para regresión, el objetivo es ajustar un hiperplano de modo que $|w \cdot x_i - b| \leq \epsilon$ para puntos de entrenamiento x_i.

9. **Ventajas y Desventajas:**

   - Ventajas: Efectivo en espacios de alta dimensión, eficaz en conjuntos de datos pequeños y versátil gracias al kernel trick.
   - Desventajas: Sensible a la escala de las características, puede ser computacionalmente costoso para grandes conjuntos de datos y requiere la elección cuidadosa de parámetros.

10. **Aplicaciones:** SVM se aplica en una variedad de problemas, como reconocimiento de escritura, clasificación de imágenes y diagnóstico médico.

2.2. Árboles de Decisión

1. **Definición:** Un árbol de decisión es una estructura jerárquica en forma de árbol que se utiliza para representar decisiones y sus posibles consecuencias. Cada nodo interno del árbol representa una prueba en una característica, cada rama representa un resultado posible de la prueba, y cada hoja representa un resultado final o una decisión.

2. **Proceso de Construcción:** El árbol se construye de manera recursiva. En cada paso, se elige la mejor característica para dividir el conjunto de datos en función de algún criterio, como la ganancia de información o la impureza de Gini. Este proceso se repite hasta que se alcanza algún criterio de parada, como la profundidad máxima del árbol o un número mínimo de puntos en una hoja.

3. **Criterios de División:** Los criterios comunes para la división incluyen:

   - **Ganancia de Información:** Mide cuánta información nueva proporciona una característica.
   - **Impureza de Gini:** Mide la probabilidad de clasificar incorrectamente un elemento si es etiquetado aleatoriamente.

4. **Ventajas y Desventajas:**

   - **Ventajas:** Fácil interpretación, no requiere normalización de datos, manejo natural de características categóricas.
   - **Desventajas:** Propenso al sobreajuste, especialmente en conjuntos de datos pequeños y ruidosos.

2.2.1. Bosques Aleatorios

1. **Definición:** Un bosque aleatorio es una colección de árboles de decisión entrenados en subconjuntos aleatorios del conjunto de datos y utilizando técnicas de agregación para mejorar la precisión y controlar el sobreajuste.

2. **Proceso de Construcción:** Se crean múltiples árboles de decisión utilizando diferentes subconjuntos del conjunto de datos de entrenamiento y características aleatorias en cada división. Luego, las predicciones de cada árbol se promedian (en regresión) o se votan (en clasificación) para obtener la predicción final del bosque.

3. **Técnica de Bagging:** La construcción de árboles en subconjuntos aleatorios del conjunto de datos se conoce como bagging (bootstrap aggregating). Esto ayuda a reducir la varianza y evitar el sobreajuste al promediar los errores.

4. **Importancia de Características:** Los bosques aleatorios proporcionan una medida de la importancia de las características, que indica cuánto contribuye cada característica a la precisión del modelo. Esto es útil para la selección de características.

5. **Ventajas y Desventajas:**

   - **Ventajas:** Reducción del sobreajuste en comparación con un solo árbol, manejo automático del sobreajuste, buen rendimiento en conjuntos de datos grandes y complejos.
   - **Desventajas:** Menos interpretables que los árboles de decisión individuales.

2.2.2. Elementos Matemáticos de Árboles de Decisión

1. **Definición:** Un árbol de decisión se representa como una función $T(x)$ que asigna una instancia x a una hoja del árbol. Cada nodo interno j realiza una prueba en una característica $f_j(x)$, y cada rama representa una condición de prueba. La estructura del árbol se define por las funciones indicadoras $I(x, j)$ que indican si la instancia x llega al nodo j.

$$T(x) = \sum_{j=1}^{J} I(x, j) \cdot C_j$$

   Donde J es el número de nodos, C_j es el valor en la hoja correspondiente al nodo j, y $I(x, j)$ es 1 si la instancia x llega al nodo j y 0 de lo contrario.

2. **Proceso de Construcción:** La construcción del árbol implica seleccionar la mejor característica f_j y el umbral t_j en cada nodo j para maximizar la ganancia de información o reducir la impureza de Gini.

$$\text{Ganancia de Información: } Gain(D, j, t) = H(D) - \frac{N_L}{N} H(D_L) - \frac{N_R}{N} H(D_R)$$

$$\text{Impureza de Gini: } Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2$$

   Donde D es el conjunto de datos en el nodo, D_L y D_R son los conjuntos de datos en los nodos izquierdo y derecho después de la división, N es el número total de instancias en D, y N_L y N_R son los números de instancias en los nodos izquierdo y derecho.

3. **Ventajas y Desventajas:**

   - **Ventajas:** Fácil interpretación, no requiere normalización de datos, manejo natural de características categóricas.
   - **Desventajas:** Propenso al sobreajuste, especialmente en conjuntos de datos pequeños y ruidosos.

2.2.3. Bosques Aleatorios

1. **Definición:** Un bosque aleatorio es una colección de B árboles de decisión $T_b(x)$, donde cada árbol se entrena en un subconjunto aleatorio de los datos de entrenamiento. La predicción se obtiene promediando (en regresión) o votando (en clasificación) las predicciones individuales de los árboles.

$$\text{Predicción del Bosque: } \hat{Y}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$$

2. **Proceso de Construcción:** Cada árbol en el bosque se construye utilizando bagging, que consiste en seleccionar aleatoriamente un subconjunto de las instancias de entrenamiento con reemplazo. Además, en cada división de nodo, se selecciona un subconjunto aleatorio de características.

$$\text{Bagging: } D_b = \{(x_i, y_i)\} \text{ con } i \sim \text{Uniforme}(1, N)$$

$$\text{Características Aleatorias: } f_j \text{ con } j \sim \text{Uniforme}(1, P)$$

3. **Importancia de Características:** La importancia de la característica f_j se mide mediante la disminución promedio en la ganancia de impureza o la reducción en el error cuadrático medio cuando se utiliza f_j para dividir los nodos a lo largo de todos los árboles.

$$\text{Importancia de } f_j = \frac{1}{B} \sum_{b=1}^{B} \sum_{j \text{ en } T_b} \text{Importancia de } f_j \text{ en } T_b$$

3. Referencias

- Libros:
  1. "The Hundred-Page Machine Learning Book"por Andriy Burkov
  2. "Machine Learning: A Probabilistic Perspective"por Kevin P. Murphy
  3. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"por Aurélien Géron
  4. "Pattern Recognition and Machine Learning"por Christopher M. Bishop
  5. "Deep Learning"por Ian Goodfellow, Yoshua Bengio y Aaron Courville
- Cursos:
  1. "Machine Learning"por Andrew Ng en Coursera
  2. .ªpplied Data Science with Python"por la Universidad de Míchigan en Coursera
  3. "Deep Learning Specialization"por Andrew Ng en Coursera
  4. "Machine Learning Crash Course"por Google
  5. "Introduction to Machine Learning"por la Universidad de Columbia en edX
- Sitios web:
  1. Kaggle
  2. GitHub
  3. Towards Data Science
  4. Machine Learning Mastery
  5. Google AI

Fuentes:

1. Machine Learning citation style [Update October 2023] - Paperpile
2. ICML2021 Template - Overleaf, Online LaTeX Editor
3. Machine Learning — Submission guidelines - Springer
4. Journal of Machine Learning Research

4. Bioinformatics: New revision

4.1. A bayesian framework for combining gene predictions - Pavlovic

Biology and biotechnology are undergoing a techonolgical revolution which is transforming research into an information-rich enterprise. A typical bacterial genome sequence is comprised of several million bases of DNA and contains several thousand genes. The human genome is approximately 3 billion bases long and it contains approximately 30,000 putative genes identified thus far.

5. Random Forest: Explanation in Mathematical Terms

Random Forest is a supervised learning algorithm used for both classification and regression problems. The main idea behind Random Forest is to build multiple decision trees during training and combine their results to obtain a more robust and accurate prediction.

Let's assume we have a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i represents the features and y_i is the target variable.

1. **Construction of Decision Trees:**
   - For each tree, a random subset of features is selected (randomly sample features), and a random subset of training data is chosen (randomly sample observations with replacement).
   - A decision tree is constructed using the subset of data and features.
   - The decision tree is represented as $h_i(x; \theta_i)$, where i denotes the tree index, x is the input feature vector, and θ_i represents the parameters of the tree.
   - At each node t of the tree, a feature j_t is selected from the random subset, and the split is determined based on minimizing impurity:

     $$\theta_{i,t} = \arg\min_{j_t, s_t} \Big[\text{Impurity}(D_t) - p_{\text{left}}\text{Impurity}(D_{\text{left}}) - p_{\text{right}}\text{Impurity}(D_{\text{right}})\Big]$$

     where D_t is the dataset at node t, D_{left} and D_{right} are the datasets in the left and right child nodes, p_{left} and p_{right} are the proportions of data in the left and right child nodes, and $\text{Impurity}(\cdot)$ is a measure of impurity, such as Gini index for classification or mean squared error for regression.
2. **Voting or Averaging:**

- For classification problems, the final prediction is obtained by majority voting among the trees:
$$H(x) = \text{mode}\{h_1(x;\theta_1), h_2(x;\theta_2), \ldots, h_n(x;\theta_n)\}$$
- For regression problems, the final prediction is the average of predictions from all trees:
$$H(x) = \frac{1}{n}\sum_{i=1}^{n} h_i(x;\theta_i)$$

3. **Variance Reduction:**
   - By building decision trees in a random manner, the variance of the model is reduced, leading to a more robust and generalizable model.

The final prediction is obtained by combining the predictions of all trees.

6. Deep Learning: Overview

Deep Learning is a subfield of machine learning that focuses on neural networks with multiple layers (deep neural networks). These networks can automatically learn hierarchical representations of data, allowing them to capture intricate patterns and features.

1. **Neural Network Representation:**
   - A neural network consists of layers of interconnected nodes (neurons) organized into an input layer, one or more hidden layers, and an output layer.
   - The input layer represents the features of the data, and each neuron in the layer processes a specific feature.
   - Hidden layers perform nonlinear transformations on the input data, learning hierarchical representations.
   - The output layer produces the final prediction based on the learned representations.
2. **Training a Neural Network:**
   - During training, the network's parameters (weights and biases) are adjusted to minimize the difference between predicted and actual outputs.
   - This optimization is typically done using backpropagation and gradient descent.
3. **Activation Functions:**
   - Activation functions introduce nonlinearity to the neural network, enabling it to learn complex patterns.
   - Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Hyperbolic Tangent (tanh).
4. **Forward Pass:**
   - The forward pass of a neural network involves computing the output of the network for a given input.

- Given an input vector x, the output y is computed by passing x through the network's layers using learned weights and biases.
- The output of each layer is computed as:

$$a^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)}$$

  where $W^{(l)}$ is the weight matrix, $a^{(l)}$ is the activation of layer l, $a^{(l-1)}$ is the activation of the previous layer, and $b^{(l)}$ is the bias vector.
- The activation function is then applied to $a^{(l)}$ to introduce nonlinearity.

5. **Loss Function:**
   - The loss function measures the difference between the predicted output and the true output.
   - Common loss functions include mean squared error for regression and cross-entropy for classification.
   - The goal during training is to minimize the loss by adjusting the network's parameters.
6. **Backpropagation:**
   - Backpropagation is a training algorithm that computes the gradient of the loss with respect to the network's parameters.
   - The gradient is used to update the parameters in the direction that reduces the loss.
   - It involves computing the gradient of the loss with respect to the output of each layer and using the chain rule to propagate these gradients backward through the network.

7. Example of Implementation in R:

7.1. Install and Load the Keras Library:

```
install.packages("keras")
library(keras)
```

7.2. Build a Simple Neural Network:

```
model <- keras_model_sequential() %>%
  layer_dense(units = 128, activation = 'relu', input_shape = c(10)) %>%
  layer_dense(units = 1, activation = 'sigmoid')

summary(model)
```

7.3. Compile and Train the Neural Network:

```
model %>% compile(
  optimizer = 'adam',
  loss = 'binary_crossentropy',
  metrics = c('accuracy')
```

```
)

# Assuming you have training data X_train and labels y_train
history <- model %>% fit(
  X_train, y_train,
  epochs = 10, batch_size = 32,
  validation_split = 0.2
)
```

7.4. Evaluate the Model:

```
# Assuming you have test data X_test and labels y_test
evaluate_result <- model %>% evaluate(X_test, y_test)
print(evaluate_result)
```

8. Impurity Measures in Random Forest

In the context of Random Forest, impurity measures play a crucial role in the construction of decision trees. The two commonly used impurity measures are the Gini index for classification and the mean squared error for regression.

8.1. Gini Index for Classification

The Gini index is a measure of impurity used in classification problems. Given a node in a decision tree that contains data points from different classes, the Gini index quantifies how often a randomly chosen data point would be incorrectly classified.

For a node t with K classes and a set of data points D_t, the Gini index ($Gini(t)$) is calculated as follows:

$$Gini(t) = 1 - \sum_{i=1}^{K} p_i^2$$

where p_i is the proportion of data points in class i at node t. A lower Gini index indicates a purer node with predominantly one class.

In the context of Random Forest, the decision tree split is determined by minimizing the weighted sum of Gini indices for the left and right child nodes. The split that results in the lowest overall Gini index is chosen.

8.2. Mean Squared Error for Regression

For regression problems, the impurity measure used is the mean squared error (MSE). Unlike classification, where impurity is related to the purity of classes in a node, regression impurity is a measure of the variability of target values within a node.

For a node t with data points D_t, the MSE ($MSE(t)$) is calculated as follows:

$$MSE(t) = \frac{1}{|D_t|} \sum_{i \in D_t} (y_i - \bar{y}_t)^2$$

where y_i is the target value of data point i, $|D_t|$ is the number of data points in node t, and $\bar{y}_t$ is the mean target value of all data points in node t.

Similar to the Gini index, in Random Forest, the decision tree split is determined by minimizing the weighted sum of MSE for the left and right child nodes.

These impurity measures guide the construction of individual decision trees within the Random Forest ensemble, contributing to the overall robustness and predictive power of the model.

9. Construcción del Árbol de Decisión

9.1. Teoría:

Un árbol de decisión es una estructura de datos que representa un conjunto de decisiones y sus posibles consecuencias. En el contexto de Random Forest, la construcción de un árbol de decisión sigue el principio de .ªprendizaje supervisado", donde el algoritmo aprende patrones a partir de un conjunto de datos etiquetado.

La construcción del árbol se realiza a través de divisiones recursivas basadas en características del conjunto de datos. Cada nodo del árbol representa una pregunta sobre una característica, y las ramas que surgen de ese nodo son las respuestas a esa pregunta. El proceso continúa hasta que se alcanza un criterio de parada, como la profundidad máxima del árbol o el número mínimo de muestras en un nodo.

9.2. Elementos Matemáticos:

1. **Función de Impureza:**

En cada nodo del árbol, se elige la característica y el umbral que minimizan la impureza en los nodos hijos resultantes. La impureza se mide mediante funciones como el Índice de Gini para clasificación o el Error Cuadrático Medio (MSE) para regresión.

Para clasificación:

$$Gini(t) = 1 - \sum_{i=1}^{K} p_i^2$$

Donde p_i es la proporción de ejemplos de la clase i en el nodo t.

Para regresión:

$$MSE(t) = \frac{1}{|D_t|} \sum_{i \in D_t} (y_i - \bar{y}_t)^2$$

Donde D_t es el conjunto de datos en el nodo t, y_i es la etiqueta del ejemplo i, y $\bar{y}_t$ es la media de las etiquetas en el nodo t.

2. **Criterio de División:**

La elección de la mejor característica y umbral se basa en la reducción de la impureza. Se busca el par (j, s) que minimiza la expresión:

$$\theta_{i,t} = \arg\min_{j,s} \left[\text{Impureza}(D_t) - p_{\text{left}} \text{Impureza}(D_{\text{left}}) - p_{\text{right}} \text{Impureza}(D_{\text{right}}) \right]$$

Donde D_t es el conjunto de datos en el nodo t, D_{left} y D_{right} son los conjuntos de datos en los nodos izquierdo y derecho después de la división, y p_{left} y p_{right} son las proporciones de datos en esos nodos.

3. **Criterios de Parada:**

Para evitar sobreajuste, se utilizan criterios de parada, como la profundidad máxima del árbol o el número mínimo de muestras requeridas para realizar una división.

La construcción de cada árbol en Random Forest implica este proceso iterativo y se repite para cada árbol en el bosque. La diversidad en la construcción de árboles se logra mediante el uso de diferentes subconjuntos aleatorios de características y datos en cada árbol. La combinación de predicciones de estos árboles mejora la generalización del modelo y su capacidad predictiva en nuevos datos.

10. Reducción de Varianza en Random Forest

10.1. Teoría:

La reducción de la varianza es uno de los objetivos clave en la construcción de árboles de decisión en Random Forest. Esta técnica busca mejorar la generalización del modelo al reducir la sensibilidad a pequeñas variaciones en los datos de entrenamiento.

La varianza se refiere a la variabilidad de las predicciones de un modelo respecto a diferentes conjuntos de datos de entrenamiento. En Random Forest, la reducción de la varianza se logra mediante dos técnicas principales: Bagging y Random Subspace.

10.1.1. Bagging (Bootstrap Aggregating):

Bagging es una técnica que consiste en entrenar múltiples modelos en diferentes subconjuntos de datos generados mediante muestreo con reemplazo (bootstrap). Cada árbol de decisión en Random Forest se entrena en un conjunto de datos ligeramente diferente, lo que introduce diversidad en los modelos.

10.1.2. Random Subspace:

Random Subspace es otra técnica utilizada para reducir la correlación entre los árboles de decisión. En lugar de usar todas las características para cada árbol, se selecciona un subconjunto aleatorio de características para entrenar cada árbol. Esto ayuda a que cada árbol se especialice en diferentes aspectos de los datos, mejorando la diversidad y reduciendo la correlación entre las predicciones.

10.2. Elementos Matemáticos:

La reducción de la varianza no se expresa directamente con fórmulas matemáticas específicas, pero los conceptos clave son fundamentales:

1. **Bagging:** - Cada árbol T_i se entrena en un conjunto de datos D_i generado por muestreo con reemplazo (bootstrap). - La predicción final se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

2. **Random Subspace:** - Cada árbol T_i se entrena utilizando un subconjunto aleatorio de características. - La predicción final se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

Estos enfoques combinados contribuyen a reducir la varianza del modelo, lo que resulta en un modelo más robusto y generalizable.

11. Votación y Promedio en Random Forest

11.1. Teoría:

La técnica de "Votación"(para problemas de clasificación) o "Promedio"(para problemas de regresión) es crucial en Random Forest para combinar las predicciones de múltiples árboles de decisión y obtener una predicción final más robusta y precisa.

11.1.1. Para Problemas de Clasificación:

En problemas de clasificación, el enfoque de votación se utiliza. Cada árbol en el bosque emite una predicción de clase, y la clase final se determina por mayoría de votos.

11.1.2. Para Problemas de Regresión:

En problemas de regresión, se utiliza un enfoque de promedio. Cada árbol realiza una predicción numérica, y la predicción final es el promedio de todas las predicciones.

11.2. Elementos Matemáticos:

1. **Votación para Clasificación:** - La predicción final para un ejemplo x se obtiene por mayoría de votos:

$$H(x) = \text{mode}\{T_1(x), T_2(x), ..., T_N(x)\}$$

2. **Promedio para Regresión:** - La predicción final para un ejemplo x se obtiene promediando las predicciones de todos los árboles:

$$H(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x)$$

Donde $T_i(x)$ representa la predicción del árbol i para el ejemplo x, y N es el número total de árboles en el bosque.

Estos enfoques de votación y promedio permiten que Random Forest combine la información de múltiples árboles de decisión, mejorando la generalización y la capacidad predictiva del modelo.

12. Resumen

Random Forest es un algoritmo de aprendizaje automático que se utiliza para resolver problemas de clasificación y regresión. En lugar de utilizar un único árbol de decisión, Random Forest construye varios árboles de decisión y los combina para obtener una predicción más precisa. Cada árbol de decisión se construye utilizando un subconjunto aleatorio de las características del conjunto de datos original. Al final, las predicciones de cada árbol se promedian para obtener una predicción final.

El algoritmo de Random Forest se basa en dos técnicas: Bagging y Random Subspace. Bagging es una técnica que se utiliza para reducir la varianza de un modelo al entrenar múltiples modelos en diferentes subconjuntos de datos. Random Subspace es una técnica que se utiliza para reducir la correlación entre los modelos al entrenar cada modelo en diferentes subconjuntos de características.

Aquí hay algunos elementos matemáticos que se utilizan en Random Forest:

- **Árbol de decisión:** Un árbol de decisión es una estructura de datos que se utiliza para modelar decisiones y sus posibles consecuencias. Cada nodo en el árbol representa una decisión, y cada rama representa una posible consecuencia de esa decisión. Los árboles de decisión se construyen utilizando un conjunto de reglas que se utilizan para tomar decisiones.
- **Bootstrap:** Bootstrap es una técnica que se utiliza para generar múltiples conjuntos de datos a partir de un conjunto de datos original. Cada conjunto de datos se genera mediante muestreo con reemplazo, lo que significa que cada elemento del conjunto de datos original tiene la misma probabilidad de ser seleccionado en cada conjunto de datos generado.
- **Out-of-Bag Error:** Out-of-Bag Error es una técnica que se utiliza para estimar el error de validación de un modelo sin la necesidad de un conjunto de datos de validación separado. El error se estima utilizando los datos que no se incluyeron en el conjunto de datos de entrenamiento para cada árbol de decisión.

12.1. Ejemplo en R: Random Forest

Aquí hay un ejemplo de cómo construir un modelo de bosque aleatorio en R:

```
# Cargamos el paquete necesario para este ejemplo
library(randomForest)

# Cargamos el conjunto de datos que deseamos utilizar
data(iris)

# Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba
trainIndex <- createDataPartition(iris$Species, p = .8, list = FALSE, times = 1)

# Entrenamos el modelo de bosque aleatorio
rf_model <- randomForest(Species ~ ., data = iris[trainIndex,])

# Realizamos predicciones en el conjunto de prueba
predictions <- predict(rf_model, iris[-trainIndex,])
```

12.2. Deep Learning

Deep Learning es un subcampo del aprendizaje automático que se centra en la creación de redes neuronales artificiales profundas. Estas redes neuronales están diseñadas para imitar el cerebro humano y son capaces de aprender patrones complejos en los datos.

Aquí hay un ejemplo de cómo construir una red neuronal profunda en R utilizando el paquete keras:

```
# Cargamos los paquetes necesarios
library(keras)
library(tensorflow)

# Cargamos el conjunto de datos que deseamos utilizar
data(iris)

# Dividimos el conjunto de datos en conjuntos de entrenamiento y prueba
trainIndex <- createDataPartition(iris$Species, p = .8, list = FALSE, times = 1)

# Creamos la red neuronal
model <- keras_model_sequential() %>%
  layer_dense(units = 4, input_shape = c(4)) %>%
  layer_activation("relu") %>%
  layer_dense(units = 3) %>%
  layer_activation("softmax")

# Compilamos la red neuronal
model %>% compile(loss = "categorical_crossentropy", optimizer = "adam", metrics = "accurac

# Entrenamos la red neuronal
history <- model %>% fit(
  x = iris[trainIndex, 1:4],
  y = to_categorical(as.numeric(iris[trainIndex, 5])),
  epochs = 100,
  batch_size = 10,
  validation_split = 0.2
)
```

13. Algoritmos de Aprendizaje Supervisado

13.1. Regresión Lineal y Logística

13.1.1. Regresión Lineal

La regresión lineal es un modelo que busca modelar la relación lineal entre una variable dependiente Y y una o más variables independientes X. El modelo se define como:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

donde:

- Y es la variable dependiente que queremos predecir.
- X es la variable independiente que utilizamos para la predicción.
- β_0 es la ordenada al origen, que representa el valor de Y cuando X es cero.
- β_1 es la pendiente de la recta, que indica cuánto cambia Y por un cambio unitario en X.
- ε es el término de error que captura la variabilidad no explicada por el modelo.

El objetivo es encontrar los valores de β_0 y β_1 que minimizan la suma de los cuadrados de los errores ε:

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 X_i))^2$$

Esto se puede hacer utilizando técnicas como el método de mínimos cuadrados.

13.1.2. Regresión Logística

La regresión logística es un modelo utilizado para problemas de clasificación binaria. Se emplea la función logística, también conocida como la función sigmoide, para transformar la salida de la regresión lineal en un valor entre 0 y 1, que se interpreta como la probabilidad de pertenecer a la clase positiva. La función sigmoide está definida como:

$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

donde:

- $P(Y = 1)$ es la probabilidad de pertenecer a la clase positiva.
- e es la base del logaritmo natural.

El modelo de regresión logística busca encontrar los valores de β_0 y β_1 que maximizan la función de verosimilitud. La función de verosimilitud es el producto de las probabilidades condicionales de observar las etiquetas de clase dados los valores de X:

$$\max_{\beta_0,\beta_1} \mathcal{L}(\beta_0, \beta_1) = \prod_{i=1}^{n} P(Y_i)^{y_i} \cdot (1 - P(Y_i))^{1-y_i}$$

donde:

- $\mathcal{L}(\beta_0, \beta_1)$ es la función de verosimilitud.
- y_i es la etiqueta de la clase para la observación i.

La regresión logística se ajusta típicamente maximizando la log-verosimilitud para obtener estimaciones de β_0 y β_1.

13.1.3. Implementacion

```
# Ejemplo de Regresión Lineal
set.seed(123)
# Crear datos de ejemplo
X <- rnorm(100)
Y <- 2 * X + rnorm(100)

# Ajustar el modelo de regresión lineal
modelo_lineal <- lm(Y ~ X)

# Imprimir resultados
summary(modelo_lineal)

# Graficar el modelo
plot(X, Y, main = "Regresión Lineal", xlab = "X", ylab = "Y")
abline(modelo_lineal, col = "red")

# Ejemplo de Regresión Logística
set.seed(123)
# Crear datos de ejemplo para clasificación binaria
X <- rnorm(100)
probabilidades <- exp(2 * X) / (1 + exp(2 * X))
Y_binario <- rbinom(100, 1, probabilidades)

# Ajustar el modelo de regresión logística
modelo_logistico <- glm(Y_binario ~ X, family = binomial)

# Imprimir resultados
summary(modelo_logistico)

# Graficar el modelo
plot(X, Y_binario, main = "Regresión Logística",
         xlab = "X", ylab = "Y", col = Y_binario + 1)
curve(predict(modelo_logistico,
           data.frame(X = x), type = "response"),
           add = TRUE, col = "red")
```

13.2. Máquinas de Soporte Vectorial (SVM)

1. **Hiperplano:** En un espacio de características n-dimensional, un hiperplano es un subespacio de dimensión $n-1$. Para un problema de clasificación binaria, un hiperplano divide el espacio en dos regiones, asignando puntos a una clase u otra.

2. **Margen:** El margen es la distancia perpendicular desde el hiperplano a los puntos más cercanos de cada clase. SVM busca el hiperplano que maximiza este margen, lo que se traduce en una mayor robustez y generalización del modelo.

3. **Vectores de Soporte:** Estos son los puntos de datos más cercanos al hiperplano y tienen un papel crucial en la definición del margen. Cambiar estos vectores de soporte afecta directamente al modelo, y son los únicos puntos que importan para la determinación del hiperplano.

4. **Función de Decisión:** La función de decisión de SVM es el hiperplano que se utiliza para clasificar nuevos puntos de datos. Dada una entrada, la función de decisión evalúa de qué lado del hiperplano cae el punto y asigna la etiqueta correspondiente.

5. **Kernel Trick:** SVM puede manejar eficientemente datos no lineales mediante el uso de funciones de kernel. Estas funciones transforman el espacio de características original en uno de mayor dimensión, permitiendo así que los datos sean separados de manera no lineal en el espacio transformado.

6. **Parámetros de SVM:**

   - **C (Parámetro de Regularización):** Controla el equilibrio entre tener un margen más amplio y clasificar correctamente los puntos de entrenamiento.
   - **Kernel:** Define la función de kernel utilizada (lineal, polinómica, radial, etc.).
   - **Gamma (para kernels no lineales):** Controla el alcance de influencia de un solo punto de datos en la decisión.

7. **Proceso de Entrenamiento:** Dado un conjunto de datos de entrenamiento etiquetado, SVM busca el hiperplano óptimo que maximiza el margen entre las clases. Esto se realiza a través de técnicas de optimización cuadrática.

8. **SVM para Regresión:** Además de la clasificación, SVM se puede utilizar para problemas de regresión. En este caso, el objetivo es ajustar un hiperplano de modo que contenga la mayor cantidad posible de puntos dentro de un margen predefinido.

9. **Ventajas y Desventajas:**

   - **Ventajas:** Efectivo en espacios de alta dimensión, eficaz en conjuntos de datos pequeños y versátil gracias al kernel trick.
   - **Desventajas:** Sensible a la escala de las características, puede ser computacionalmente costoso para grandes conjuntos de datos y requiere la elección cuidadosa de parámetros.

10. **Aplicaciones:** SVM se utiliza en una variedad de campos, como reconocimiento de escritura, clasificación de imágenes, diagnóstico médico, entre otros.

13.2.1. Elementos Matemáticos de las SVM

1. **Hiperplano:** Un hiperplano se define como $w \cdot x - b = 0$, donde w es el vector de pesos, x es el vector de entrada, y b es el sesgo.

2. **Margen:** El margen M entre un hiperplano y un punto x_i se define como $M = \frac{1}{\|w\|}|w \cdot x_i - b|$.

3. **Vectores de Soporte:** Los vectores de soporte son los puntos x_i que cumplen la condición $|w \cdot x_i - b| = 1/\|w\|$.

4. **Función de Decisión:** La función de decisión es $f(x) = w \cdot x - b$. Si $f(x) > 0$, el punto x se clasifica como clase 1; si $f(x) < 0$, se clasifica como clase -1.

5. **Kernel Trick:** La función de kernel $K(x, x')$ representa el producto escalar en un espacio de características de mayor dimensión. Ejemplos comunes incluyen el kernel lineal ($K(x, x') = x \cdot x'$), kernel polinómico ($K(x, x') = (x \cdot x' + 1)^d$), y kernel radial ($K(x, x') = \exp(-\gamma \|x - x'\|^2)$).

6. **Parámetros de SVM:**

   - C (Parámetro de Regularización): Se introduce en la función de pérdida para controlar el equilibrio entre tener un margen más amplio y clasificar correctamente los puntos de entrenamiento.
   - w (Vector de Pesos): Aprende durante el entrenamiento y define la orientación del hiperplano.
   - b (Sesgo): Parámetro de ajuste del hiperplano.
   - γ (para kernels no lineales): Controla el alcance de influencia de un solo punto de datos en la decisión.

7. **Proceso de Entrenamiento:** El proceso de entrenamiento implica la minimización de la función de pérdida, que incluye el término de regularización $C\|w\|^2$ y la función de pérdida hinge.

8. **SVM para Regresión:** Para regresión, el objetivo es ajustar un hiperplano de modo que $|w \cdot x_i - b| \leq \epsilon$ para puntos de entrenamiento x_i.

9. **Ventajas y Desventajas:**

   - Ventajas: Efectivo en espacios de alta dimensión, eficaz en conjuntos de datos pequeños y versátil gracias al kernel trick.
   - Desventajas: Sensible a la escala de las características, puede ser computacionalmente costoso para grandes conjuntos de datos y requiere la elección cuidadosa de parámetros.

10. **Aplicaciones:** SVM se aplica en una variedad de problemas, como reconocimiento de escritura, clasificación de imágenes y diagnóstico médico.

13.3. Árboles de Decisión y Bosques Aleatorios

1. **Definición:** Un árbol de decisión es una estructura jerárquica en forma de árbol que se utiliza para representar decisiones y sus posibles consecuencias. Cada nodo interno del árbol representa una prueba en una característica, cada rama representa un resultado posible de la prueba, y cada hoja representa un resultado final o una decisión.

2. **Proceso de Construcción:** El árbol se construye de manera recursiva. En cada paso, se elige la mejor característica para dividir el conjunto de datos en función de algún criterio, como la ganancia de información o la impureza de Gini. Este proceso se repite hasta que se alcanza algún criterio de parada, como la profundidad máxima del árbol o un número mínimo de puntos en una hoja.

3. **Criterios de División:** Los criterios comunes para la división incluyen:

   - **Ganancia de Información:** Mide cuánta información nueva proporciona una característica.

- **Impureza de Gini:** Mide la probabilidad de clasificar incorrectamente un elemento si es etiquetado aleatoriamente.

4. **Ventajas y Desventajas:**
   - **Ventajas:** Fácil interpretación, no requiere normalización de datos, manejo natural de características categóricas.
   - **Desventajas:** Propenso al sobreajuste, especialmente en conjuntos de datos pequeños y ruidosos.

13.3.1. Bosques Aleatorios

1. **Definición:** Un bosque aleatorio es una colección de árboles de decisión entrenados en subconjuntos aleatorios del conjunto de datos y utilizando técnicas de agregación para mejorar la precisión y controlar el sobreajuste.

2. **Proceso de Construcción:** Se crean múltiples árboles de decisión utilizando diferentes subconjuntos del conjunto de datos de entrenamiento y características aleatorias en cada división. Luego, las predicciones de cada árbol se promedian (en regresión) o se votan (en clasificación) para obtener la predicción final del bosque.

3. **Técnica de Bagging:** La construcción de árboles en subconjuntos aleatorios del conjunto de datos se conoce como bagging (bootstrap aggregating). Esto ayuda a reducir la varianza y evitar el sobreajuste al promediar los errores.

4. **Importancia de Características:** Los bosques aleatorios proporcionan una medida de la importancia de las características, que indica cuánto contribuye cada característica a la precisión del modelo. Esto es útil para la selección de características.

5. **Ventajas y Desventajas:**
   - **Ventajas:** Reducción del sobreajuste en comparación con un solo árbol, manejo automático del sobreajuste, buen rendimiento en conjuntos de datos grandes y complejos.
   - **Desventajas:** Menos interpretables que los árboles de decisión individuales.

13.3.2. Elementos Matemáticos de Árboles de Decisión

1. **Definición:** Un árbol de decisión se representa como una función $T(x)$ que asigna una instancia x a una hoja del árbol. Cada nodo interno j realiza una prueba en una característica $f_j(x)$, y cada rama representa una condición de prueba. La estructura del árbol se define por las funciones indicadoras $I(x, j)$ que indican si la instancia x llega al nodo j.

$$T(x) = \sum_{j=1}^{J} I(x, j) \cdot C_j$$

Donde J es el número de nodos, C_j es el valor en la hoja correspondiente al nodo j, y $I(x, j)$ es 1 si la instancia x llega al nodo j y 0 de lo contrario.

2. **Proceso de Construcción:** La construcción del árbol implica seleccionar la mejor característica f_j y el umbral t_j en cada nodo j para maximizar la ganancia de información o reducir la impureza de Gini.

$$\text{Ganancia de Información: } Gain(D, j, t) = H(D) - \frac{N_L}{N} H(D_L) - \frac{N_R}{N} H(D_R)$$

$$\text{Impureza de Gini: } Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2$$

Donde D es el conjunto de datos en el nodo, D_L y D_R son los conjuntos de datos en los nodos izquierdo y derecho después de la división, N es el número total de instancias en D, y N_L y N_R son los números de instancias en los nodos izquierdo y derecho.

3. **Ventajas y Desventajas:**
   - **Ventajas:** Fácil interpretación, no requiere normalización de datos, manejo natural de características categóricas.
   - **Desventajas:** Propenso al sobreajuste, especialmente en conjuntos de datos pequeños y ruidosos.

13.3.3. Bosques Aleatorios

1. **Definición:** Un bosque aleatorio es una colección de B árboles de decisión $T_b(x)$, donde cada árbol se entrena en un subconjunto aleatorio de los datos de entrenamiento. La predicción se obtiene promediando (en regresión) o votando (en clasificación) las predicciones individuales de los árboles.

$$\text{Predicción del Bosque: } \hat{Y}(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$$

2. **Proceso de Construcción:** Cada árbol en el bosque se construye utilizando bagging, que consiste en seleccionar aleatoriamente un subconjunto de las instancias de entrenamiento con reemplazo. Además, en cada división de nodo, se selecciona un subconjunto aleatorio de características.

$$\text{Bagging: } D_b = \{(x_i, y_i)\} \text{ con } i \sim \text{Uniforme}(1, N)$$

$$\text{Características Aleatorias: } f_j \text{ con } j \sim \text{Uniforme}(1, P)$$

3. **Importancia de Características:** La importancia de la característica f_j se mide mediante la disminución promedio en la ganancia de impureza o la reducción en el error cuadrático medio cuando se utiliza f_j para dividir los nodos a lo largo de todos los árboles.

$$\text{Importancia de } f_j = \frac{1}{B} \sum_{b=1}^{B} \sum_{j \text{ en } T_b} \text{Importancia de } f_j \text{ en } T_b$$

13.4. Redes Neuronales

14. Aprendizaje No Supervisado

14.1. K-Means y Clustering Jerárquico

14.2. Análisis de Componentes Principales (PCA)

14.3. Algoritmos de Asociación

14.4. Mapas Autoorganizados (SOM)

15. Evaluación de Modelos y Métricas

15.1. Precisión, Sensibilidad, Especificidad

15.2. Curvas ROC y Área Bajo la Curva (AUC-ROC)

15.3. Matriz de Confusión

15.4. Validación Cruzada

16. Preprocesamiento de Datos

16.1. Normalización y Estandarización

16.2. Manejo de Datos Faltantes

16.3. Ingeniería de Características

16.4. Selección de Características

17. Optimización de Modelos

17.1. Hiperparámetros y Búsqueda en Cuadrícula

17.2. Optimización Bayesiana

17.3. Regularización

17.4. Redes Neuronales Convolucionales (CNN) y Recurrentes (RNN)

18. Aprendizaje por Refuerzo

18.1. Q-Learning

18.2. Algoritmos de Políticas

18.3. Exploración y Explotación

18.4. Funciones de Valor

19. Ética en el Machine Learning

19.1. Sesgo y Equidad

19.2. Transparencia y Explicabilidad

19.3. Privacidad y Seguridad

19.4. Responsabilidad en el Despliegue de Modelos

20. Redes Neuronales Profundas

20.1. Perceptrones Multicapa (MLP)

20.2. Funciones de Activación: ReLU, Sigmoid, Tanh

20.3. Backpropagation

20.4. Regularización en Redes Neuronales

21. Redes Neuronales Convolucionales (CNN)

21.1. Convolutional Layers

21.2. Pooling Layers

21.3. Transfer Learning con CNN

21.4. Aplicaciones en Visión por Computadora

22. Redes Neuronales Recurrentes (RNN)

22.1. Arquitecturas de RNN

22.2. Long Short-Term Memory (LSTM)

22.3. Gated Recurrent Unit (GRU)

22.4. Aplicaciones en Procesamiento de Lenguaje Natural

23. Redes Generativas

23.1. Generative Adversarial Networks (GAN)

23.2. Variational Autoencoders (VAE)

23.3. Aplicaciones en Generación de Imágenes y Texto

24. Transferencia de Aprendizaje en Deep Learning

24.1. Fine-Tuning de Modelos Preentrenados

24.2. Domain Adaptation

24.3. Modelos Preentrenados como BERT, GPT

25. Técnicas Avanzadas

25.1. Normalización por Lotes (Batch Normalization)

25.2. Dropout

25.3. Redes Siamesas

25.4. Redes Neuronales Adversarias Condicionales (cGAN)

26. Herramientas y Frameworks

26.1. TensorFlow

26.2. PyTorch

26.3. Keras

26.4. TensorBoard para Visualización