Árboles, Regresión y SVM: Resumen

1. Árboles de Decisión

- Los **árboles de decisión** son una técnica de aprendizaje automático por inducción que permite identificar conceptos a partir de las características de un conjunto de ejemplos. La información se organiza jerárquicamente en forma de grafo dirigido con nodos y arcos, donde los nodos corresponden a preguntas o tests sobre los ejemplos.
- Un árbol de decisión se construye haciendo preguntas sobre las características de los ejemplos y clasificándolos según la respuesta, actuando como un **clasificador**. Las opciones de clasificación son excluyentes entre sí, llevando a una única conclusión.
- La construcción requiere un **conjunto de entrenamiento**, una **representación simbólica del conocimiento** (atributos y valores), un **algoritmo de aprendizaje** (clasificación o regresión), y un **método de evaluación**.
- Un árbol tiene un **nodo raíz**, **nodos intermedios** que pueden ser raíces de subárboles, y **hojas** que representan los conceptos extraídos (una clase específica). Cada nodo intermedio y la raíz tienen **separadores** que preguntan sobre las características, determinando los nodos sucesores.
- El **atributo seleccionado como separador** debe generar subárboles simples y una clasificación concreta. Es crucial determinar los atributos importantes y su orden de uso.
- El algoritmo general implica **calcular la calidad del nodo**, y si no es suficiente, **calcular el mejor atributo separador** para dividir el conjunto en subconjuntos de mayor calidad. Este proceso se repite hasta cumplir un **criterio de parada**.
- Los **criterios de selección** basados en la calidad del nodo incluyen el **índice de Entropía**, el **índice de Gini**, y la **ganancia de información**. El índice de entropía mide la incertidumbre, mientras que el índice de Gini mide la probabilidad de no sacar dos registros con el mismo valor para la variable objetivo.
- La **importancia de las características** es su contribución relativa a las predicciones, calculada estimando la reducción agregada y normalizada del error de predicción para cada división. Las variables más importantes tienen un mayor impacto en las predicciones.
- Los **criterios de parada** evitan el sobreaprendizaje e incluyen el máximo número de ramas por nodo, el número mínimo de observaciones por nodo final, el número mínimo de observaciones para dividir un nodo, y no encontrar variables suficientemente discriminantes.
- La **poda (pruning)** es recomendable para reducir el árbol, simplificándolo al eliminar nodos redundantes. Se basa en un algoritmo de mínimo coste/complejidad considerando el coste de clasificación errónea, el número de hojas, y el coste de introducir una nueva hoja.
- Los modelos basados en árboles de decisión dividen los datos en particiones maximizando la ganancia de información (clasificación) o minimizando el error cuadrático medio (regresión). El proceso de partición es recursivo, formando una jerarquía con predicados asociados.
- Los **árboles de regresión** se utilizan para predecir valores numéricos continuos, dividiendo los datos para lograr la mayor reducción de varianza. La predicción se basa en el promedio de los valores de la variable objetivo en el nodo terminal. Los **árboles de modelos** son una extensión donde cada nodo contiene un modelo (lineal u otro) ajustado a los datos de ese nodo, permitiendo capturar relaciones más complejas.
- El **sobreajuste** es un problema común en árboles de decisión, por lo que se utilizan criterios de parada y poda para evitarlo.

- Los árboles de decisión tienen ventajas como la **facilidad de entendimiento e interpretación**, ser **no paramétricos**, y **manejar diversos tipos de datos**. Sin embargo, son propensos al **sobreajuste**, son **sensibles a variaciones en los datos**, y pueden tener problemas con **datos desbalanceados**.
- **Bagging** y **Random Forests** utilizan árboles individuales con cierto nivel de aleatoriedad, mientras que **Boosted Trees** construyen secuencialmente árboles que intentan corregir los errores del modelo anterior.

2. Selección de Modelos

- Seleccionar el mejor modelo de Machine Learning implica **comparar distintos modelos** y determinar cuál tiene el mejor rendimiento según los datos y objetivos.
- El proceso incluye **comprender el problema y los datos**, **preprocesar los datos** (manejo de nulos, codificación, escalado, selección de características), **seleccionar modelos candidatos** (para clasificación y regresión).
- Para cada modelo candidato, se realiza **entrenamiento y evaluación** dividiendo los datos en entrenamiento y prueba, utilizando **validación cruzada** y comparando **métricas de rendimiento** adecuadas (precisión, recall, F1-score, RMSE, R^2, etc.).
- La **optimización de hiperparámetros** se realiza para mejorar el modelo seleccionado utilizando técnicas como **Grid Search**, **Random Search**, y métodos más avanzados como **Optuna** o **Bayesian Optimization**.
- Después de la optimización, se realiza la **comparación de modelos** utilizando curvas ROC-AUC, curvas de error de validación, matrices de confusión, considerando el tiempo de entrenamiento y predicción, y la interpretabilidad.
- La **interpretación del modelo y la explicabilidad** son importantes una vez seleccionado el modelo final, utilizando técnicas como **Feature Importance**, **SHAP**, y **LIME**.
- Finalmente, se realiza la **implementación y despliegue** del modelo óptimo.
- La elección entre **Bagging**, **Random Forest**, y **Boosting** depende de si el objetivo principal es reducir la varianza (Bagging, Random Forest) o el sesgo (Boosting), la sensibilidad al ruido, la necesidad de interpretabilidad, y los recursos computacionales disponibles. **Random Forest** a menudo se considera un buen punto de partida.
- La **regresión lineal sin regularización** es preferible cuando el número de muestras es mucho mayor que el de características, no hay multicolinealidad, hay pocas variables irrelevantes o ruido bajo, y cuando se necesita interpretar los coeficientes tal como son.

3. Regresión

- La **regresión lineal** es un enfoque supervisado para predecir valores cuantitativos, asumiendo una relación lineal entre la observación X y la respuesta Y. Se define por la ecuación $Y = \beta 0 + \beta 1X + \varepsilon$.
- Los coeficientes $\beta 0$ (intersección) y $\beta 1$ (pendiente) se estiman mediante el método de **mínimos cuadrados ordinarios (OLS)**, que minimiza la suma de los errores cuadráticos (RSS).
- Un modelo de regresión lineal básico debe cumplir ciertas **hipótesis**, como esperanza matemática nula de los errores, homocedasticidad, incorrelación de los errores, regresores no estocásticos, independencia lineal, número de observaciones mayor que el de parámetros ($T > k + 1$), y normalidad de los errores.

- Las **limitaciones de los modelos lineales** se abordan con **modelos de regresión no lineales** como la **regresión polinomial**, las **funciones escalonadas**, los **splines de regresión**, y las **funciones base**.
- La **regresión logística** se utiliza para predecir la probabilidad de pertenencia a una clase en problemas de clasificación binaria, utilizando la **función sigmoide** o logística. Los parámetros se estiman mediante **máxima verosimilitud**. Los coeficientes indican la correlación con la clase predicha.
- El proceso de regresión logística incluye la búsqueda de variables correlacionadas, discretización de variables numéricas, entrenamiento del modelo inicial (sin interacciones) y con interacciones, entrenamiento del modelo jerárquico, elección del mejor modelo y evaluación.
- `LinearRegression()` en Scikit-Learn resuelve la regresión lineal analíticamente usando la fórmula cerrada de mínimos cuadrados (OLS). Es rápido para datos pequeños o medianos.
- `SGDRegressor()` en Scikit-Learn utiliza el **descenso por gradiente estocástico (SGD)**, siendo útil para datasets muy grandes. Tiene hiperparámetros importantes como `loss, penalty, alpha, learning_rate, max_iter, tol, early_stopping, etc..`
- La **regularización** es crucial para evitar el sobreajuste, añadiendo una penalización a los coeficientes. Los métodos principales son **Ridge (L2)**, **Lasso (L1)**, y **Elastic Net (L1 + L2)**. Ridge reduce el sobreajuste en multicolinealidad, Lasso realiza selección de características al hacer algunos coeficientes cero, y Elastic Net combina beneficios de ambos.
- **Escalar los datos** no es estrictamente necesario para `LinearRegression()` sin regularización en términos de predicción, pero puede mejorar la estabilidad numérica y la interpretación de los coeficientes, especialmente si hay grandes diferencias en la escala de las variables. **Es altamente recomendable escalar los datos cuando se utiliza regularización (Ridge, Lasso, Elastic Net)** y con `PolynomialFeatures()` debido a la sensibilidad a la escala de los coeficientes penalizados.

4. Support Vector Machine

- La **Máquina de Vectores de Soporte (SVM)** es un algoritmo de aprendizaje estadístico potente para clasificación, considerado un buen clasificador de "caja negra".
- El **Clasificador de Margen Máximo** se basa en el concepto de **hiperplano** para separar datos linealmente separables, utilizando el hiperplano con la mayor distancia mínima a las observaciones de entrenamiento (**vectores de soporte**).
- Los **Clasificadores de Soporte Vectorial** extienden esta idea a datos casi separables, permitiendo algunas observaciones dentro del margen o en el lado incorrecto del hiperplano (**margen suave**) para mayor robustez.
- Las **Máquinas de Soporte Vectorial** utilizan **kernels** para transformar los datos a un espacio de características de mayor dimensión donde se puede encontrar un hiperplano de separación lineal para datos no linealmente separables en el espacio original.
- Los **tipos de kernel** comunes incluyen el **kernel polinomial** y el **kernel de base radial (RBF)**.
- El SVM se ha extendido para **clasificación multiclase**, **detección de novedades**, y **regresión**.
- En Scikit-Learn, las implementaciones principales son **SVC**, **NuSVC**, y **LinearSVC**. Los parámetros habituales incluyen el **kernel** (lineal, poli, rbf), la **regularización** (**C**), y el parámetro

del kernel **gamma**. **C** controla el equilibrio entre el margen y el error de clasificación, mientras que **gamma** modela la parametrización del kernel.
- Las **ventajas principales** del SVM son que el modelo se basa en pocos vectores de soporte (predicción rápida y poco uso de memoria), funciona bien con datos de alta dimensión, y es muy versátil gracias a los kernels.
- Sus **desventajas principales** son el coste computacional que puede ser alto con un gran número de muestras, la falta de una interpretación probabilística directa, y la gran dependencia de la elección adecuada del parámetro de suavización **C**.

5. Descenso del Gradiente

El descenso del gradiente (o *gradient descent*) es un **algoritmo clave dentro del campo del *machine learning* y se encuentra en el corazón de la gran mayoría de los sistemas de inteligencia artificial** que se desarrollan hoy en día. Su objetivo principal es **encontrar el punto mínimo de una función de coste**, la cual mide el error de un modelo para diferentes combinaciones de sus parámetros.

En modelos con funciones de coste convexas, encontrar el mínimo es sencillo mediante el método de mínimos cuadrados. Sin embargo, **la diversidad de modelos y funciones de coste en el *machine learning* nos obliga a encontrar una solución para las funciones no convexas**. Estas funciones no convexas presentan el problema de poder tener **múltiples mínimos locales que no son el mínimo global**.

El descenso del gradiente aborda este problema aprovechando la información que proporciona el cálculo de la derivada (o derivadas parciales en funciones multidimensionales). **La derivada en un punto indica la pendiente de la función en ese punto**. En un contexto tridimensional, como un terreno con colinas y valles, la derivada nos diría la inclinación del terreno.

La lógica del algoritmo es la siguiente:

- Se comienza con unos **parámetros iniciales del modelo establecidos de forma aleatoria**, lo que equivaldría a estar en un punto cualquiera de ese terreno.
- En la posición actual, se **evalúa la pendiente de la función de coste calculando su derivada (o gradiente)**. El gradiente es un vector que indica la dirección hacia donde la pendiente asciende.
- Como queremos descender, tomamos el **sentido opuesto al gradiente** para actualizar nuestros parámetros.
- Avanzamos una **distancia determinada en esa dirección** y nos detenemos en una nueva posición. La distancia que avanzamos está controlada por un parámetro llamado **ratio de aprendizaje**.
- Se **repite este proceso iterativamente** hasta llegar a una zona donde movernos ya no suponga una variación notable del coste, es decir, la pendiente es próxima a cero, lo que probablemente indica que hemos llegado a un mínimo local.

El **ratio de aprendizaje** es crucial, ya que define cuánto afecta el gradiente a la actualización de los parámetros en cada iteración o cuánto avanzamos en cada paso. Un ratio de aprendizaje **muy pequeño** puede hacer que el algoritmo tarde mucho en converger y pueda quedar atrapado en mínimos locales. Un ratio de aprendizaje **muy elevado** puede hacer que los pasos sean demasiado grandes, impidiendo que el algoritmo converja al mínimo y causando un bucle infinito.

En resumen, el descenso del gradiente es un **método iterativo de optimización que utiliza el gradiente de la función de coste para encontrar sus mínimos**, siendo especialmente relevante para funciones no convexas donde existen múltiples mínimos. La correcta elección del ratio de aprendizaje es fundamental para su funcionamiento eficiente.