1. **K-Nearist Neighbor (KNN)**

Algoritma k-Nearest Neighbor (k-NN) adalah sebuah metode dalam data mining dan machine learning yang digunakan untuk klasifikasi dan regresi. Metode ini bekerja dengan menghitung jarak antara data yang baru dengan data latihan yang ada, dan kemudian memilih k data latihan terdekat sebagai tetangga terdekatnya. Berikut adalah langkah-langkah dalam algoritma k-NN:

1. Menentukan jumlah tetangga terdekat (k) yang akan digunakan dalam proses klasifikasi atau regresi.
2. Menghitung jarak antara data yang baru (query) dengan setiap data latihan yang ada. Jarak ini biasanya dihitung menggunakan metrik Euclidean, Manhattan, atau metrik jarak lainnya.
3. Memilih k data latihan dengan jarak terdekat dari data query. Ini adalah tetangga terdekat.
4. Jika tujuan adalah klasifikasi, maka menggunakan mayoritas suara (majority voting) dari label tetangga terdekat untuk memprediksi label data query. Label yang paling sering muncul di antara tetangga terdekat akan menjadi label prediksi.
5. Jika tujuan adalah regresi, maka menggunakan rata-rata nilai target dari tetangga terdekat untuk memprediksi nilai target data query.

Algoritma k-NN juga memiliki beberapa hal yang perlu dipertimbangkan, antara lain:

1. Skala data: Penting untuk melakukan normalisasi atau standarisasi data jika fitur-fiturnya memiliki skala yang berbeda. Hal ini mencegah fitur-fitur dengan skala besar mendominasi perhitungan jarak.
2. Jumlah tetangga (k): Jumlah tetangga yang dipilih dapat memiliki pengaruh terhadap hasil prediksi. Jika k terlalu kecil, algoritma dapat menjadi sensitif terhadap noise. Jika k terlalu besar, algoritma dapat menjadi kurang sensitif terhadap variasi dalam data.
3. Pemilihan metrik jarak: Pemilihan metrik jarak juga dapat mempengaruhi hasil prediksi. Metrik Euclidean adalah yang paling umum digunakan, tetapi ada juga metrik lain seperti metrik Manhattan, Chebyshev, atau metrik jarak Minkowski.

Algoritma k-NN relatif sederhana dan mudah dipahami, tetapi dapat memberikan hasil yang baik terutama untuk dataset yang tidak terlalu besar. Namun, perlu diingat bahwa algoritma ini memiliki kompleksitas waktu yang tinggi saat digunakan pada dataset yang sangat besar, karena membutuhkan perhitungan jarak dengan setiap data latihan.

1. **Support Vector Machine (SVM)**

Algoritma Support Vector Machine (SVM) adalah metode pembelajaran mesin yang digunakan untuk klasifikasi dan regresi. Tujuannya adalah untuk menemukan hyperplane yang memaksimalkan jarak antara kelas-kelas yang berbeda dalam ruang fitur.

Berikut adalah langkah-langkah umum dalam algoritma SVM:

1. Persiapan data: Mulai dengan mengumpulkan dan mempersiapkan data yang akan digunakan untuk pelatihan SVM. Pastikan data tersebut dalam format numerik dan lakukan normalisasi jika diperlukan.
2. Pembentukan ruang fitur: Identifikasi fitur-fitur yang akan digunakan dalam proses pembelajaran. Pilih fitur-fitur yang relevan dan dapat membedakan kelas-kelas yang berbeda.
3. Pemilihan kernel: Pilih kernel yang sesuai dengan data Anda. Kernel mengubah data ke dalam ruang fitur yang lebih tinggi agar dapat diklasifikasikan dengan baik. Beberapa kernel yang umum digunakan termasuk linear, polinomial, radial basis function (RBF), dan sigmoid.
4. Pembentukan model: Latih model SVM dengan menggunakan data latih. Model SVM mencoba menemukan hyperplane yang memaksimalkan margin antara kelas-kelas yang berbeda dalam ruang fitur. Dalam kasus SVM linier, hyperplane ini merupakan garis atau bidang yang memisahkan kelas-kelas.
5. Optimasi parameter: Selama pelatihan, parameter SVM seperti C (regularization parameter) dan gamma (untuk kernel RBF) perlu dioptimalkan. Hal ini dapat dilakukan dengan menggunakan teknik penalaan silang (cross-validation) untuk mencari kombinasi parameter terbaik yang menghasilkan kinerja yang optimal pada data validasi.
6. Klasifikasi atau regresi: Setelah model SVM dilatih, Anda dapat menggunakannya untuk melakukan klasifikasi atau regresi pada data baru. Model ini akan menentukan kelas mana yang paling mungkin berdasarkan pada posisi data baru dalam ruang fitur.
7. Evaluasi model: Terakhir, evaluasi model SVM untuk mengukur kinerjanya. Gunakan metrik evaluasi seperti akurasi, presisi, recall, atau F1-score untuk mengevaluasi sejauh mana model dapat mengklasifikasikan data dengan benar.

Algoritma SVM dapat diterapkan untuk masalah klasifikasi biner, klasifikasi multikelas, dan regresi dengan sedikit modifikasi pada langkah-langkah di atas. SVM juga dapat digunakan dalam konteks pembelajaran mendukung yang tidak sepenuhnya terarah (unsupervised learning) dengan menggunakan metode seperti SVM One-Class atau Support Vector Clustering (SVC).

Perlu diingat bahwa SVM merupakan algoritma yang kuat namun juga memiliki beberapa kelemahan, seperti sensitif terhadap skala data dan kebutuhan perhatian yang tinggi terhadap pemilihan parameter yang tepat.

1. **Random Forest**

Algoritma Random Forest adalah sebuah metode yang digunakan dalam pembelajaran mesin untuk melakukan klasifikasi, regresi, dan tugas-tugas lainnya. Metode ini menggunakan konsep ensemble learning, di mana beberapa model pembelajaran mesin yang lemah digabungkan untuk membentuk model yang lebih kuat.

Berikut adalah langkah-langkah utama dalam algoritma Random Forest:

1. Pilih jumlah pohon (n\_tree) yang akan dibentuk dalam Random Forest.
2. Untuk setiap pohon dalam Random Forest:

- Acak sampel sebagian data training dari dataset yang ada (bootstrap sampling) dengan mengambil sampel dengan pengembalian.

- Pada setiap simpul dalam pohon, pilih secara acak sejumlah fitur yang akan digunakan untuk melakukan pemisahan simpul (splitting).

- Pilih fitur yang memberikan pemisahan yang paling baik berdasarkan suatu kriteria (misalnya, indeks Gini untuk klasifikasi atau penurunan kesalahan kuadrat rata-rata untuk regresi).

- Pisahkan simpul menjadi dua anak simpul berdasarkan fitur dan ambang pemisahan (threshold) yang dipilih.

- Teruskan proses splitting hingga mencapai kriteria penghentian (misalnya, jumlah sampel minimum di setiap simpul atau kedalaman maksimum pohon).

1. Setiap pohon dalam Random Forest akan menghasilkan prediksi berdasarkan mayoritas voting (klasifikasi) atau rata-rata (regresi) dari prediksi-prediksi individu pohon.
2. Untuk melakukan prediksi pada data baru, setiap pohon dalam Random Forest memberikan prediksi dan hasil akhir diambil berdasarkan mayoritas voting (klasifikasi) atau rata-rata (regresi) dari prediksi-prediksi individu pohon.

Keuntungan dari menggunakan algoritma Random Forest adalah:

1. Mampu menangani dataset yang besar dengan banyak fitur.
2. Tahan terhadap overfitting karena penggunaan bootstrap sampling dan random feature selection.
3. Mampu mengatasi masalah multicollinearity dalam dataset.
4. Memberikan perkiraan kepentingan fitur untuk mengevaluasi relevansi fitur dalam prediksi.

Demikianlah langkah-langkah umum dalam algoritma Random Forest. Penting untuk dicatat bahwa implementasi yang tepat dapat bervariasi tergantung pada perpustakaan atau framework yang digunakan.

1. **Multilayer Perceptron (MLP)**

Algoritma Multilayer Perceptron (MLP) adalah salah satu jenis arsitektur jaringan saraf tiruan (neural network) yang paling umum digunakan. MLP terdiri dari beberapa lapisan (layer) yang saling terhubung, yaitu lapisan input (input layer), lapisan tersembunyi (hidden layers), dan lapisan keluaran (output layer). Setiap lapisan terdiri dari sejumlah unit pemrosesan (neuron) yang menerima input, melakukan komputasi, dan mengirimkan output ke unit-unit di lapisan berikutnya.

Berikut adalah algoritma dasar untuk melatih MLP menggunakan metode pembelajaran berbasis gradien (gradient-based learning):

1. Inisialisasi Bobot dan Bias: Setiap unit di dalam MLP memiliki bobot (weight) dan bias (bias) yang harus diinisialisasi secara acak dengan nilai kecil.

2. Forward Propagation: Proses dimulai dengan mengirimkan input ke lapisan input. Setiap unit di lapisan tersembunyi dan lapisan keluaran menghitung nilai input yang diterimanya dengan memperhitungkan bobot dan bias. Kemudian, nilai output dari setiap unit dikirim ke unit-unit di lapisan berikutnya hingga mencapai lapisan keluaran.

3. Perhitungan Error: Setelah mendapatkan output dari MLP, perhitungkan selisih antara output yang dihasilkan dengan output yang diharapkan. Error ini digunakan untuk mengukur sejauh mana MLP melakukan prediksi yang akurat.

4. Backward Propagation: Langkah ini melibatkan propagasi balik error dari lapisan keluaran ke lapisan tersembunyi. Error di setiap unit di lapisan keluaran digunakan untuk memperbarui bobot dan bias yang menghubungkan lapisan tersembunyi dan keluaran. Kemudian, error di setiap unit di lapisan tersembunyi dihitung berdasarkan error di lapisan berikutnya dan digunakan untuk memperbarui bobot dan bias yang menghubungkan lapisan tersembunyi dengan lapisan sebelumnya. Proses ini dilakukan secara berulang-ulang hingga error yang dihasilkan mencapai tingkat yang dapat diterima.

5. Update Bobot dan Bias: Setelah menghitung error di seluruh MLP, langkah terakhir adalah memperbarui bobot dan bias pada setiap unit. Perbarui bobot dan bias dengan menggunakan algoritma optimasi seperti stochastic gradient descent (SGD) atau varian-varian lainnya.

6. Ulangi Langkah 2-5: Ulangi langkah-langkah forward propagation, perhitungan error, backward propagation, dan update bobot dan bias secara berulang-ulang sampai kriteria berhenti terpenuhi. Kriteria berhenti bisa berupa jumlah iterasi maksimum atau mencapai tingkat akurasi yang diinginkan.

7. Evaluasi dan Prediksi: Setelah pelatihan selesai, gunakan MLP yang telah dilatih untuk melakukan evaluasi dan prediksi pada data baru.

Itulah algoritma dasar untuk melatih dan menggunakan MLP. Perlu diingat bahwa ada banyak variasi dan penyesuaian yang dapat dilakukan pada MLP, seperti penggunaan fungsi aktivasi yang berbeda