Nama : Ghina Khoerunnisa
Program : Data Scientist Intern

Classification II

Naive Bayes Classifier => Berdasarkan Bayes Theorem seperti pada formula di bawah,

$$p(h|D) = (p(D|h).p(h)) / p(D)$$

Yang mana p(h|D) merupakan posterior, p(D|h) merupakan likelihood, p(h) merupakan prior, dan p(D) merupakan evidence. Algoritma Naive Bayes memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya. Algoritma ini disebut naive karena menganggap tidak ada dependensi di variabel-variabelnya. Naive Bayes classifier memiliki akurasi dan kecepatan tinggi pada kumpulan data besar. Naive Bayes Classifier bekerja dengan cara yang pertama menghitung prior probability, kemudian menghitung likelihood, lalu dengan kedua hasil tersebut hitung posteriornya dan bandingkan class mana yang probabilitasnya lebih tinggi maka data tersebut masuk ke class tersebut. Naive Bayes Classifier dapat diimplementasikan dengan menggunakan library scikit-learn yaitu GaussianNB() dari sklearn.naive_bayes..

```
3
4  #Create a Gaussian Classifier
5  model = GaussianNB()
6
7  # Train the model using the training sets
8  model.fit(features,label)
9
```

Decision Tree Classifier => Berdasarkan pohon keputusan, di mana internal node mewakili feature(atau attribute), branch mewakili decision rule, dan setiap leaf node mewakili outcome. Cara kerja algoritma ini adalah pertama memilih atribut terbaik menggunakan Attribute Selection Measures (ASM) untuk split record. Kemudian jadikan atribut sebagai decision node dan bagi dataset menjadi subset yang lebih kecil. Lalu ulangi proses tersebut berulang-ulang hingga tercapai suatu kondisi (Semua tupel memiliki nilai atribut yang sama, tidak ada lagi atribut yang tersisa, tidak ada lagi instances). Pada Decision Tree ASM yang dapat di tuning ada gini impurity (index yang mendeskripsikan classification error), entropy (uncertainties), dan information gain (entropy sebelum - entropy sesudah). Jika menggunakan gini maka memilih gini index yang rendah yang berarti errornya kecil. Dengan menggunakan decision tree yang terlalu dalam dan leaves nya banyak maka akan menyebabkan model overfit dan begitu sebaliknya jika terlalu sedikit maka akan underfit. Decision Tree Classifier dapat diimplementasikan dengan menggunakan library scikit-learn yaitu DecisionTreeClassifier() dari sklearn.tree.

```
1  # Create Decision Tree classifer object
2  clf = DecisionTreeClassifier()
3
4  # Train Decision Tree Classifer
5  clf = clf.fit(X_train,y_train)
6
```

Random Forest Classifier => Gabungan dari beberapa decision tree yang mana keputusan akhirnya berdasarkan voting dari seluruh decision tree yang dibangun. Random forest termasuk ke dalam ensemble method bagian yang bagging. Bagging maksudnya adalah high variance learner digabungkan untuk menurunkan variance nya, considerably low bias.

Random forest bekerja dengan cara yaitu membagi dataset menjadi beberapa sampel yang lebih kecil secara random, kemudian bangun decision tree untuk setiap sampel hingga mendapatkan prediksi dari setiap decision tree. Lalu terakhir lakukan voting terhadap prediksi yang paling banyak dan prediksi itulah yang menjadi prediksi finalnya. Selain membuat model, dengan random forest kita juga bisa melakukan feature importance. Features importance menunjukkan importance relatif atau kontribusi setiap fitur dalam prediksi. Secara otomatis menghitung skor relevansi setiap fitur dalam fase training kemudian menurunkan relevansi sehingga jumlah semua skor adalah 1. Random forest dapat diimplementasikan dengan menggunakan library scikit-learn yaitu RandomForestClassifier() dari sklearn.ensemble.

```
#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)
```

Support Vector Machines => SVM membangun hyperplane (decision plane yang memisahkan antara sekumpulan objek yang memiliki kelas yang berbeda) dengan margin semaksimal mungkin yang paling baik membagi dataset menjadi menjadi beberapa kelas. Cara kerjanya, pertama generate hyperplane-hyperplane yang memisahkan kelas dengan cara terbaik. Kemudian pilih hyperplane yang memiliki classification error terkecil atau pilih hyperplane yang tepat dengan segregasi maksimum dari salah satu titik data terdekat. Algoritma SVM diimplementasikan dalam praktiknya menggunakan kernel. Kernel mengubah input data space menjadi bentuk yang diperlukan. Disini terdapat 3 kernel yang dapat digunakan yaitu linear kernel, polynomial kernel, dan radial basis function kernel. Support vector machines classifier dapat diimplementasikan dengan menggunakan library scikit-learn yaitu SCV() dari sklearn.svm.

```
#Create a svm Classifier
clf = SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)
```

Cross Validation => Dilakukan pada data training karena untuk menghindari adanya data leakage dan untuk mengetahui seberapa generalize model yang dibangun.