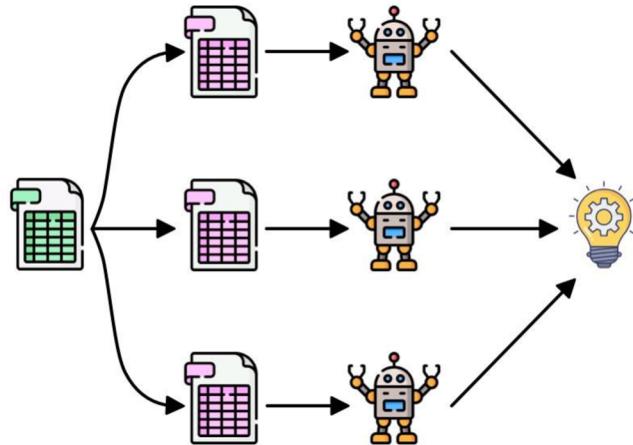




Implementasi Metode Bagging Dalam Random Forest

Oleh: Muhammad Mirza Fahmi

Metode Bagging



Teknik bagging diimplementasikan dengan cara **mengagregat** beberapa estimator yang menggunakan model yang sama dilatih dengan **subhimpunan data latih yang berbeda**. Cara ini disebut sebagai *bootstrapping*.

Bootstrapping adalah metode pengambilan sampel dengan pengganti dari data

Bootstrapping

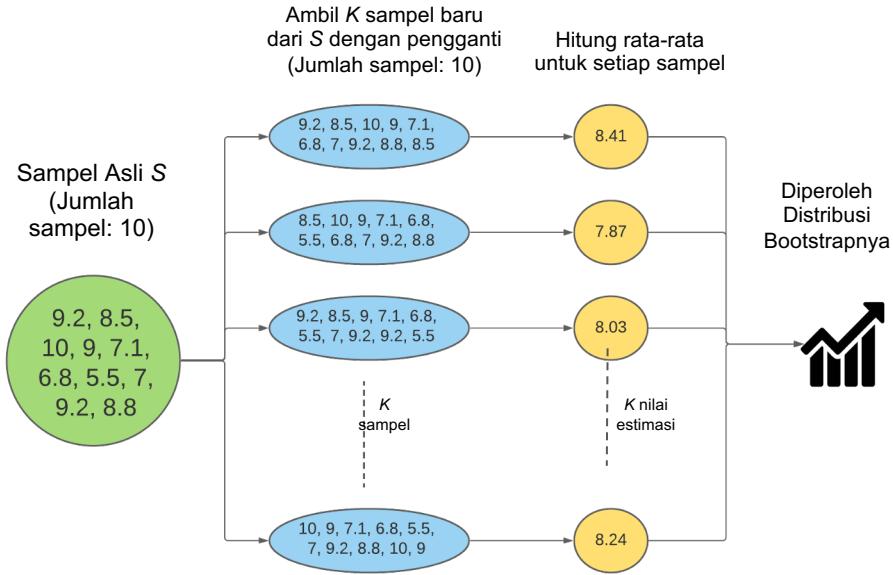
Contoh kasus, semisal kita ingin mengestimasi rata-rata nilai matematika siswa kelas 3 SMP dan kita hanya diberikan sampel data berikut:

$$S = \{9.2, 8.5, 10, 9, 7.1, 6.8, 5.5, 7, 9.2, 8.8\}$$

$$\text{Grade} = \frac{9.2 + 8.5 + 10 + 9 + 7.1 + 6.8 + 5.5 + 7 + 9.2 + 8.8}{10} = 8.11$$

Jika kita mengestimasi dengan nilai-rata, diperoleh sebagai berikut: Tapi dengan hanya 10 data, kurang merepresentasikan populasi data. Jadi kita tidak bisa yakin dengan hasil estimasinya dan perlu sebuah margin error.

Solusinya adalah bootstrapping yaitu membuat beberapa sampel baru dari data sampel sebelumnya, kemudian hitung rata-rata masing-masing sampel. Dari hasil tersebut, dapat kita peroleh distribusi bootstrap, dan selanjutnya dapat kita peroleh error standar untuk hasil estimasinya.



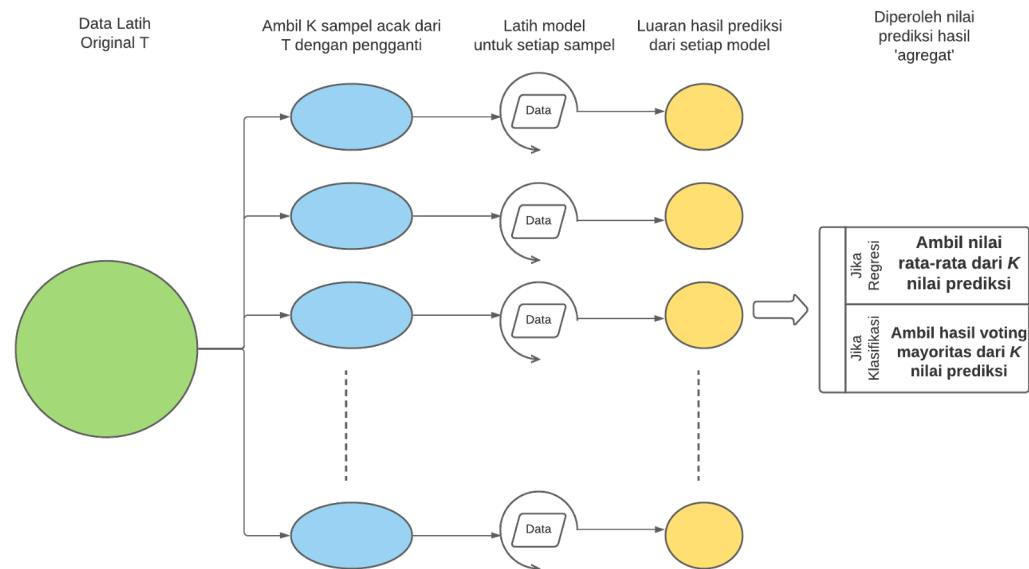
Bootstrapping adalah metode pengambilan sampel dengan pengganti dari data asli.

Ilustrasi Metode Bagging

Proses bagging dalam machine learning pun serupa, yaitu ketika data latih dibuat sejumlah K sampel secara acak. Kemudian setiap sampel tersebut, dibuat model kemudian hasil prediksi masing-masingnya kemudian di aggregasi.

Untuk permasalahan klasifikasi, ambil hasil terbaiknya dengan cara voting dari nilai-nilai agregasinya.

Untuk permasalahan regresi, ambil hasil terbaiknya dengan cara menghitung rata-ratanya.



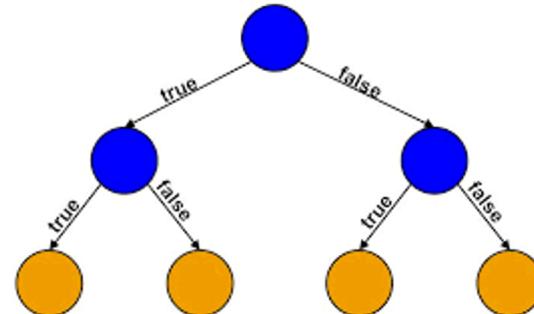


Pengenalan Random Forest

Review: Decision Tree

Decision tree adalah algoritma berbasis struktur pohon atau hirarki. Algoritma ini mudah dipahami karena meniru cara berfikir manusia.

Nama lain dari *decision tree* adalah CART (*Classification and Regression Tree*).

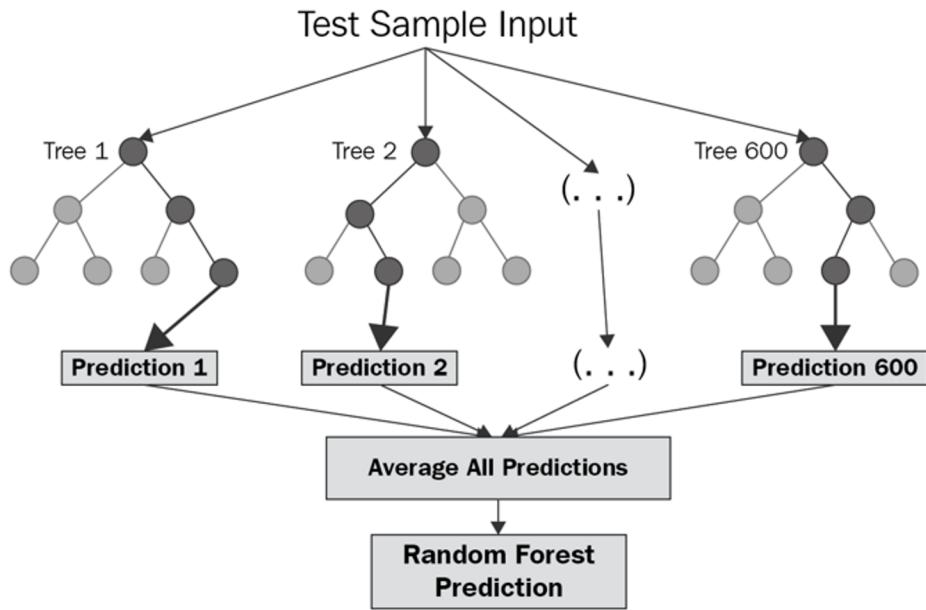


Keuntungan

- Mudah diinterpretasi dan dipahami.
- Hanya butuh sedikit data untuk dilatih.
- Bisa untuk masalah klasifikasi dan regresi.
- Toleran terhadap nilai yang hilang.

Kekurangan

- Cukup rentan terhadap over fitting terhadap data latih.
- Sensitif terhadap penciran.
- Pembelajar lemah.



Random Forest

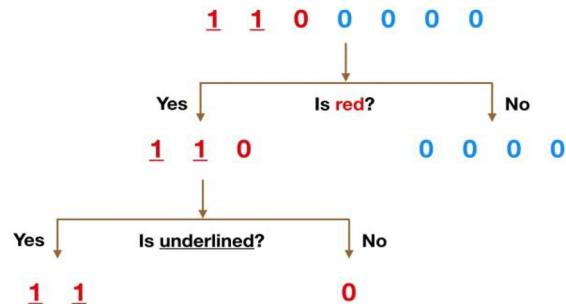
Salah satu kekurangan *Decision Tree* yaitu sangat rentan terhadap *overfitting* dan memperoleh hasil prediksi yang kurang baik.

Karenanya untuk menghindari hal tersebut, algoritma ini ditambah dengan metode ensemble, sehingga diperoleh hasil prediksi yang lebih mendekati nilai sesungguhnya.

Algoritma ini disebut sebagai **Random Forest**.

Tipe-tipe model Random Forest

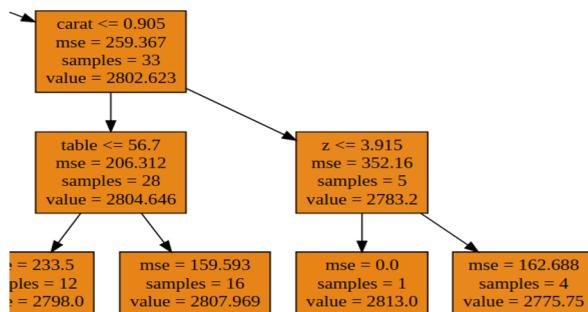
Classification



```

from sklearn.ensemble import RandomForestClassifier
X = [[0, 0], [1, 1]]
Y = [0, 1]
clf = RandomForestClassifier(n_estimators=10)
clf = clf.fit(X, Y)
  
```

Regression



```

from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
X, y = make_regression(n_features=4, n_informative=2,
                       random_state=0, shuffle=False)
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(X, y)

print(regr.predict([[0, 0, 0, 0]]))
  
```

RF From Scratch (Cont'd)

__init__()

Konstruktor, menyimpan nilai hyperparameter berupa jumlah pohon, jumlah minimum sampel yang dipecah, dan kedalaman maksimum. Juga menyimpan setiap individu *Decision Tree* yang telah dilatih.

_sample(X, y)

Fungsi yang mengimplementasikan bootstrap sampling.

```
def __init__(self, num_trees=25, min_samples_split=2, max_depth=5):
    self.num_trees = num_trees
    self.min_samples_split = min_samples_split
    self.max_depth = max_depth
    # Will store individually trained decision trees
    self.decision_trees = []

@staticmethod
def _sample(X, y):
    """
    Helper function used for bootstrap sampling.

    :param X: np.array, features
    :param y: np.array, target
    :return: tuple (sample of features, sample of target)
    """

    n_rows, n_cols = X.shape
    # Sample with replacement
    samples = np.random.choice(a=n_rows, size=n_rows, replace=True)
    return X[samples], y[samples]
```

RF From Scratch (Cont'd)

fit(X, y)

Fungsi untuk melatih model mesin pembelajar.

```
def fit(self, X, y):
    """
    Trains a Random Forest classifier.

    :param X: np.array, features
    :param y: np.array, target
    :return: None
    """

    # Reset
    if len(self.decision_trees) > 0:
        self.decision_trees = []

    # Build each tree of the forest
    num_built = 0
    while num_built < self.num_trees:
        try:
            clf = DecisionTree(
                min_samples_split=self.min_samples_split,
                max_depth=self.max_depth
            )
            # Obtain data sample
            _X, _y = self._sample(X, y)
            # Train
            clf.fit(_X, _y)
            # Save the classifier
            self.decision_trees.append(clf)
            num_built += 1
        except Exception as e:
            continue
```

predict(X)

Fungsi untuk membuat prediksi.

```
def predict(self, X):
    """
    Predicts class labels for new data instances.

    :param X: np.array, new instances to predict
    :return:
    """

    # Make predictions with every tree in the forest
    y = []
    for tree in self.decision_trees:
        y.append(tree.predict(X))

    # Reshape so we can find the most common value
    y = np.swapaxes(a=y, axis1=0, axis2=1)

    # Use majority voting for the final prediction
    predictions = []
    for preds in y:
        counter = Counter(x)
        predictions.append(counter.most_common(1)[0][0])
    return predictions
```

Quiz

1. Bagaimana random forest menghitung hasil prediksi?
2. Apa perbedaan mendasar dari decision tree dengan random forest?
3. Metode apa yang digunakan random forest Ketika melakukan sampling?