Chairul Imam
1608107010040
Final Lab Data Mining

Dataset Preprocessing

Langkah dan screenshot yang dimuat pada dokumen ini adalah snippet utama dari notebook yang lebih lengkap. Source code yang lebih lengkap dapat diakses pada bagian **Source Code**.

Ganti ? dengan NaN

Karakter '?' yang terdapat didalam instance diganti dengan **NaN** untuk mempermudah pembersihan data. Instance dengan NaN dapat dihapus dengan fungsi **dropna** pada library **Pandas**.

Encode Categorical Data

Dataset adult memiliki banyak feature bertipe categorical atau object. Untuk mengencode feature tersebut saya menggunakan **OrdinalEncoder** dari library **sklearn.preprocessing**. Feature bertipe categorical antara lain: workclass, education, marital-status, occupation, relationship, race, sex, native-country, dan salary.

Setelah diencode, dataframe training dan testing disimpan. Dan untuk membuat file **adult-complete.csv**, saya mengconcat dataframe training dan testing.

Transaksi dengan item NONE dihapus dengan mencari index transaksi tersebut. Dengan informasi index transaksi, fungsi drop akan menghapus transaksi tersebut.

```
# save index where Item == NONE
index_of_none = df[ df.Item == 'NONE' ].index

# dimension of  instance where item == NONE
index_of_none.shape
```

```
(786,)
```

```
# drop instance according to index_of_none
df.drop(index=index_of_none,
        inplace=True)
```

Membangun Model

Teknik Train_Test_Split

KNN

```
# bangun model
model = KNeighborsClassifier(n_neighbors=23,
                             weights='distance',
                             metric='manhattan')

# fit data training
model.fit(X_train, y_train)

# test model
y_pred = model.predict(X_test)
```

```
from sklearn.metrics import accuracy_score, classification_report

# print akurasi
print(f"Akurasi: {accuracy_score(y_test, y_pred):.4f}")

# print precision, recall & F-1 Score
print(classification_report(y_pred=y_pred,
                            y_true=y_test))
```

```
Akurasi: 0.8547
              precision    recall  f1-score   support

         0.0       0.89      0.93      0.90      6745
         1.0       0.75      0.65      0.69      2300

    accuracy                           0.85      9045
   macro avg       0.82      0.79      0.80      9045
weighted avg       0.85      0.85      0.85      9045
```

Dari teknik train_test_split, KNN menghasilkan akurasi 0.8547.

SVM

```
# bangun model
model = SVC(C=10, gamma=0.01, kernel='rbf')

# fit data training
model.fit(X_train, y_train)

# test model
y_pred = model.predict(X_test)
```

```
# print akurasi
print(f"Akurasi: {accuracy_score(y_test, y_pred):.4f}")

# print precision, recall & F-1 Score
print(classification_report(y_pred=y_pred,
                            y_true=y_test))
```

```
Akurasi: 0.8533
              precision    recall  f1-score   support

         0.0       0.87      0.94      0.91      6745
         1.0       0.77      0.60      0.67      2300

    accuracy                           0.85      9045
   macro avg       0.82      0.77      0.79      9045
weighted avg       0.85      0.85      0.85      9045
```

Dari teknik train_test_split, SVM menghasilkan akurasi 0.8533.

Teknik Using Data Testing

KNN

```
# bangun model
model = KNeighborsClassifier(n_neighbors=23,
                             weights='distance',
                             metric='manhattan')

# fit data training
model.fit(X_train, y_train)

# test model
y_pred = model.predict(X_test)
```

```
# print akurasi
print(f"Akurasi: {accuracy_score(y_test, y_pred):.4f}")

# print precision, recall & F-1 Score
print(classification_report(y_pred=y_pred,
                            y_true=y_test))
```

```
Akurasi: 0.8536
              precision    recall  f1-score   support

         0.0       0.89      0.92      0.90     11360
         1.0       0.73      0.64      0.68      3700

    accuracy                           0.85     15060
   macro avg       0.81      0.78      0.79     15060
weighted avg       0.85      0.85      0.85     15060
```

Dari teknik data testing, KNN menghasilkan akurasi 0.8536.

SVM

```
# bangun model
model = SVC(C=10, gamma=0.01, kernel='rbf')

# fit data training
model.fit(X_train, y_train)

# test model
y_pred = model.predict(X_test)
```

```
# print akurasi
print(f"Akurasi: {accuracy_score(y_test, y_pred):.4f}")

# print precision, recall & F-1 Score
print(classification_report(y_pred=y_pred,
                            y_true=y_test))
```

```
Akurasi: 0.8562
              precision    recall  f1-score   support

         0.0       0.88      0.94      0.91     11360
         1.0       0.77      0.59      0.67      3700

    accuracy                           0.86     15060
   macro avg       0.82      0.77      0.79     15060
weighted avg       0.85      0.86      0.85     15060
```

Dari teknik data tersting, SVM menghasilkan akurasi 0.8562.

Kesimpulan

Dengan menggunakan teknik train_test_split, **KNN menghasilkan akurasi yang lebih tinggi dibanding SVM dengan nilai 0.8547 dan 0.8533** untuk masing-masing model. Untuk model SVM saya tidak melakukan GridSearch karena ukuran dataset yang terlalu besar sehingga membutuhkan waktu training yang lama.

Dengan menggunakan teknik data testing, **SVM menghasilkan akurasi yang lebih tinggi dibanding KNN dengan nilai 0.8562 dan 0.8536** untuk masing-masing model. Untuk model SVM saya tidak melakukan GridSearch karena ukuran dataset yang terlalu besar sehingga membutuhkan waktu training yang lama.

Model KNN dan SVM dibangun dengan parameter yang sama agar dapat dibandingkan performanya dalam klasifikasi menggunakan 2 teknik yang berbeda, train_test_split dan data testing.

Source Code

Untuk mengakses dan mencoba source code, silakan mengakses pada URL berikut: https://github.com/chairul-imam/Data-Mining-and-Machine-Learning/tree/main/Lab/Final

Referensi

https://medium.com/@erikgreenj/k-neighbors-classifier-with-gridsearchcv-basics-3c445ddeb657