

Nama Dosen : Teguh Iman Hermanto, M.Kom
 Mata Kuliah : Machine Learning 2
 Pembahasan : ANN Diabetes
 Pokok Pemb : Membuat model klasifikasi penyakit diabetes menggunakan ANN



UCI MACHINE LEARNING AND 1 COLLABORATOR · UPDATED 8 YEARS AGO

4480

<> Code

Download



Pima Indians Diabetes Database

Predict the onset of diabetes based on diagnostic measures



Data Card Code (3448) Discussion (54) Suggestions (0)

About Dataset

Context

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Usability

8.82

License

CC0: Public Domain

Equivalent Dataset

View benchmark on OpenML



```
1 !pip install kaggle
```



```
1 !kaggle datasets download -d uciml/pima-indians-diabetes-database
```



```
1 !unzip pima-indians-diabetes-database.zip -d content
```



```
1 import pandas as pd
2 import numpy as np
3
4 import tensorflow as tf
5 from tensorflow.keras.models import Sequential
6 from tensorflow.keras.layers import Dense
7 from tensorflow.keras.utils import plot_model
8
9 from sklearn.model_selection import train_test_split
10 from sklearn.preprocessing import StandardScaler
11 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

PREPROCESSING DATA



```
1 data = pd.read_csv('content/diabetes.csv')
```



```
1 data.head()
```



```
1 # Pisahkan fitur (X) dan target (y)
2 X = data.drop("Outcome", axis=1)
3 y = data["Outcome"]
```



```
1 # Split dataset menjadi training dan testing set
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```



```
1 # Normalisasi data
2 scaler = StandardScaler()
3 X_train = scaler.fit_transform(X_train)
4 X_test = scaler.transform(X_test)
```

MODELING



```
1 # Membangun model ANN
2 model = Sequential()
3 model.add(Dense(16, activation='relu', input_shape=(X_train.shape[1],))) # Hidden layer
4 model.add(Dense(8, activation='relu')) # Hidden layer kedua
5 model.add(Dense(1, activation='sigmoid')) # Output layer untuk klasifikasi biner
```



```
1 # Compile model
2 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```



```
1 # Plot arsitektur model
2 plot_model(model, show_shapes=True, show_layer_names=True)
```



```
1 # Melatih model
2 history = model.fit(X_train, y_train, epochs=50, batch_size=16, validation_split=0.2, verbose=1)
```

TESTING MODEL



```
1 # Prediksi pada data test
2 y_pred_prob = model.predict(X_test)
3 y_pred = (y_pred_prob > 0.5).astype(int) # Konversi probabilitas ke kelas biner
```



```
1 # Menghitung akurasi
2 accuracy = accuracy_score(y_test, y_pred)
3 print(f'Akurasi: {accuracy * 100:.2f}%')
```



```
1 # Menampilkan confusion matrix dan classification report
2 print("\nConfusion Matrix:")
3 print(confusion_matrix(y_test, y_pred))
```



```
1 print("\nClassification Report:")
2 print(classification_report(y_test, y_pred))
```

SIMULASI MODEL



```
1 # Contoh input data baru (8 fitur)
2 new_data = np.array([[6, 148, 72, 35, 0, 33.6, 0.627, 50]]) # Contoh data pasien
3
4 # Normalisasi input data baru
5 new_data_scaled = scaler.transform(new_data)
6
7 # Prediksi
8 prediction_prob = model.predict(new_data_scaled)
9 prediction = (prediction_prob > 0.5).astype(int)
10
11 # Hasil prediksi
12 if prediction[0] == 1:
13     print("Pasien diprediksi terkena diabetes.")
14 else:
15     print("Pasien diprediksi tidak terkena diabetes.")
```



```
1 model.save("diabetes_model.h5")
```

OPTIMASI MODEL



```
1 from tensorflow.keras.layers import Dropout
2 from tensorflow.keras.regularizers import l2
```



```
1 # Membangun model ANN Optimasi
2 model = Sequential()
3 model.add(Dense(64, activation='relu', input_shape=X_train.shape[1:], kernel_regularizer=l2(0.01))) # Hidden layer
4 Dropout(0.3)
5 model.add(Dense(32, activation='relu', kernel_regularizer=l2(0.01))) # Hidden layer kedua
6 Dropout(0.3)
7 model.add(Dense(16, activation='relu', kernel_regularizer=l2(0.01)))
8 model.add(Dense(1, activation='sigmoid')) # Output layer untuk klasifikasi biner
```



```
1 # Compile model
2 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```



```
1 # Plot arsitektur model
2 plot_model(model, show_shapes=True, show_layer_names=True)
```



```
1 # Melatih model
2 from tensorflow.keras.callbacks import EarlyStopping
3
4 # Callback untuk early stopping
5 early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
6
7 # Melatih model
8 history = model.fit(
9     X_train, y_train,
10    epochs=100, # Lebih banyak epoch
11    batch_size=16,
12    validation_split=0.2,
13    callbacks=[early_stopping],
14    verbose=1
15 )
```