LAPORAN UJIAN TENGAH SEMESTER

MATA KULIAH *MACHINE LEARNING*

TK-45-GAB-G04

Pengolahan Dataset RegresiUTSTelkom menggunakan Regression Model

Disusun untuk memenuhi UTS mata kuliah Machine Learning

di Program Studi S1 Teknik Komputer

Disusun oleh:

MUHAMMAD RAFINDHA ASLAM

1103213080

FAKULTAS TEKNIK ELEKTRO

UNIVERSITAS TELKOM

BANDUNG

2024

1. Latar Belakang

Model regresi memiliki peran penting dalam memprediksi nilai target berdasarkan fitur tertentu, khususnya dalam data science dan machine learning. Pemilihan model yang optimal memerlukan evaluasi parameter model menggunakan teknik seperti hyperparameter tuning untuk meningkatkan akurasi prediksi. Pada tugas ini, dilakukan tuning hyperparameter untuk empat model regresi: Polynomial Regression, Decision Tree, k-Nearest Neighbors (k-NN), dan XGBoost.

Model ini diterapkan pada dataset yang berisi data numerik dengan target kolom 2001. Dataset ini memiliki beberapa nilai kosong (missing values) yang perlu ditangani melalui preprocessing. Dataset ini mencakup berbagai fitur sebagai prediktor, dengan target 2001 merepresentasikan variabel dependen yang akan diprediksi menggunakan model regresi.

2. Tujuan

1) Melakukan hyperparameter tuning untuk menemukan parameter terbaik dari setiap model regresi.
2) Mengevaluasi performa model menggunakan metrik seperti Mean Squared Error (MSE), Mean Absolute Error (MAE), dan R^2 Score.
3) Membandingkan hasil tuning untuk menentukan model yang paling optimal berdasarkan dataset yang diberikan.

3. Metode

1) Preprocessing Data:

- Mengatasi nilai kosong (missing values) menggunakan SimpleImputer dengan strategi imputasi rata-rata (mean).
- Membagi dataset menjadi training (80%) dan testing (20%).
- Sampling 20% data training untuk efisiensi memori saat tuning hyperparameter.

2) Hyperparameter Tuning:

- Menggunakan RandomizedSearchCV untuk mengurangi beban komputasi dengan parameter grid yang lebih sederhana.
- Menggunakan validasi silang (cross-validation) sebanyak 2 folds.

3) Evaluasi:

- Mengevaluasi model berdasarkan MSE, MAE, dan R^2 Score.
- Membandingkan performa model untuk memilih model terbaik.

4. Code

1) Bagian 1: Exploratory Data Analysis (EDA)

```
# Import necessary libraries for EDA
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv('RegresiUTSTelkom.csv')  # Ensure the file path is
correct in Google Colab

# Display basic information
print("Dataset Information:")
print(data.info())

print("\nDataset Description:")
print(data.describe())

# Calculate the number of rows and columns for the layout
num_cols = data.shape[1]  # Get the number of columns
num_rows = (num_cols + 3) // 4  # Calculate rows, ensuring enough space
(+3 to round up)


# Plotting distributions of each feature
# Adjust the layout to accommodate all columns
data.hist(bins=15, figsize=(15, 4 * num_rows), layout=(num_rows, 4))
plt.suptitle("Feature Distributions")
plt.tight_layout()  # Adjust subplot params for a tight layout
plt.show()

# Correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

Bagian 2: Preprocessing dan Pipeline Setup

```
# Import necessary libraries for data processing, pipelines, models,
and metrics
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from xgboost import XGBRegressor
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
import pandas as pd

# Load your dataset
data = pd.read_csv('RegresiUTSTelkom.csv')  # Ensure file path is
correct

# Split the data into features and target
print("Column names in the dataset:", data.columns)
X = data.drop('2001', axis=1)  # Replace '2001' with the actual target
column name if necessary
y = data['2001']  # Target variable

# Split dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Define model pipelines
poly_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('poly_features', PolynomialFeatures(degree=2)),
    ('regressor', DecisionTreeRegressor())
])

dt_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', DecisionTreeRegressor())
])

knn_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', KNeighborsRegressor())
])

xgb_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', XGBRegressor())
])

print("Pipelines created successfully.")
```

Bagian 3: Polynomial Regression Tuning

```
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
```

```
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import RandomizedSearchCV

# Define pipeline with imputer for Polynomial Regression
poly_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),  # Handle missing
values by imputing the mean
    ('scaler', StandardScaler()),
    ('poly_features', PolynomialFeatures(degree=2)),
    ('regressor', DecisionTreeRegressor())
])

# Polynomial Regression Hyperparameter Tuning
poly_params = {
    'poly_features__degree': [2],  # Only degree 2
    'regressor__max_depth': [None, 10]
}

poly_search = RandomizedSearchCV(
    poly_pipeline,
    param_distributions=poly_params,
    n_iter=2,  # Match the total space of parameters
    cv=2,
    scoring='neg_mean_squared_error',
    n_jobs=1,
    verbose=2,
    random_state=42
)

# Fit the pipeline on the training data sample
poly_search.fit(X_train_sample, y_train_sample)

# Store results
results['Polynomial Regression'] = {
    "Best Parameters": poly_search.best_params_,
    "Evaluation": evaluate_model(poly_search.best_estimator_, X_test,
y_test)
}

# Print results
print("Best Polynomial Regression Parameters:", results['Polynomial
Regression']['Best Parameters'])
print("Evaluation Metrics:", results['Polynomial
Regression']['Evaluation'])
```

Bagian 4: k-Nearest Neighbors Tuning

```
from sklearn.impute import SimpleImputer

# Update k-NN pipeline with imputer
knn_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),  # Handle missing
values by imputing the mean
    ('scaler', StandardScaler()),
    ('regressor', KNeighborsRegressor())
])
```

Bagian 5: XGBoost Tuning

```
# k-Nearest Neighbors Hyperparameter Tuning
knn_params = {
    'regressor__n_neighbors': [3, 5, 7],
    'regressor__weights': ['uniform', 'distance']
}
knn_search = RandomizedSearchCV(
    knn_pipeline,
    param_distributions=knn_params,
    n_iter=5,
    cv=2,
    scoring='neg_mean_squared_error',
    n_jobs=1,
    verbose=2,
    random_state=42
)
knn_search.fit(X_train_sample, y_train_sample)
results['k-Nearest Neighbors'] = {
    "Best Parameters": knn_search.best_params_,
    "Evaluation": evaluate_model(knn_search.best_estimator_, X_test,
y_test)
}

# XGBoost Hyperparameter Tuning
xgb_params = {
    'regressor__n_estimators': [50, 100, 200],
    'regressor__max_depth': [3, 6, 10],
    'regressor__learning_rate': [0.01, 0.1, 0.2]
}
xgb_search = RandomizedSearchCV(
    xgb_pipeline,
    param_distributions=xgb_params,
    n_iter=5,
    cv=2,
```

```
    scoring='neg_mean_squared_error',
    n_jobs=1,
    verbose=2,
    random_state=42
)
xgb_search.fit(X_train_sample, y_train_sample)
results['XGBoost'] = {
    "Best Parameters": xgb_search.best_params_,
    "Evaluation": evaluate_model(xgb_search.best_estimator_, X_test,
y_test)
}

# Display the results for each model
for model_name, result in results.items():
    print(f"\nModel: {model_name}")
    print("Best Parameters:", result["Best Parameters"])
    print("Evaluation Metrics:")
    for metric, value in result["Evaluation"].items():
        print(f" - {metric}: {value}")
```

5. Analisis

1) Polynomial Regression: Performa sangat rendah (R^2 = 0.064, MSE = 112.83), menunjukkan model ini tidak efektif dalam menjelaskan variabilitas data.
2) k-Nearest Neighbors: Sedikit lebih baik dari Polynomial Regression (R^2 = 0.196, MSE = 96.89), tetapi hasilnya masih kurang optimal.
3) XGBoost: Memberikan hasil terbaik dengan R^2 = 0.317 dan MSE = 82.34, menunjukkan kemampuan menangkap pola data lebih baik dibandingkan model lainnya.

6. Kesimpulan

Berdasarkan evaluasi terhadap tiga model, XGBoost adalah pilihan terbaik untuk tugas ini karena memiliki akurasi tertinggi dengan R^2 sebesar 0.317 dan MSE terendah, menunjukkan kemampuannya yang superior dalam menangkap pola data dibandingkan Polynomial Regression dan k-Nearest Neighbors.