DAFTAR PUSTAKA

Adyah Widiarni, M. (2021). Penerapan Algoritma Support Vector Regression untuk Prediksi Jumlah Pasien Covid-19 di Provinsi Riau . *Building of Informatics, Technology and Science (BITS)*, 8.

Angelo Rossini, D. A. (2024). Unsupervised K-means Analysis of Tuberculosis Data in Brazil: Identifying High Prevalence States and Temporal Trends. *ScienceDirect*, 8.

Firman Akbar, H. W. (2022). Implementasi Algoritma Decision Tree C4.5 dan Support Vector Regression untuk Prediksi Penyakit Stroke . *Indonesian Journal of Machine Learning and Computer Science*, 7.

Manyi Xu, W. L. (2024). DDCM: A Computational Strategy for Drug Repositioning Based on Support-Vector Regression Algorithm . *International Journal of Molcular Science*, 25.

Muhammad Hafi Isfahan Isnan, A. I. (2024). Sistem Informasi Penyebaran Penyakit Tuberculosis Paru Di Puskesmas Karang Rejo Dengan Metode K-Means Clustering Berbasis . *Jurnal Penerapan Teknologi Informasi*, 11.

Ozlem Akay, M. T. (2021). Use of the Support Vector Regression in Medical Data Analysis. *Experimental and Applied Medical Science.*, 15.

Prasetyo Arta Kusuma, A. U. (2022). Deteksi Penyebaran Penyakit Tuberkulosis dengan Algoritma K-Means Clustering Menggunakan Rapid Miner. *Jurnal Teknlogi Informatika dan Komputer MH. Thamrin*, 14.

Shynta Ayu Dwi Darmawan, K. (2024). Penerapan Metode K-Means Clustering Dan simple Moving Averange Untuk Memprediksi Jenis Penyakit Di Provinsi Jawa Timur . *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, 10.

Sumawiyah Hsb, I. H. (2024). Peramalan Jumlah Kasus Tuberkulosis Di Rumah Sakit Umum Haji Medan Dengan Metode Support Regression – Particle Swarm Optimization. *Jurnal Penelitian Matematika dan Pendidikan Matematika*, 10.

Wahyudi, A. A. (2024). Deteksi Penyakit Tuberculosis Berbasis Chest XRay Menggunakan Metode Convolution Neural Network Pada Platform Android. 71.

LAMPIRAN

Publikasi Jurnal (On Review)

Active Submissions

ACTIVE ARCHIVE

ID	MM-DD SUBMIT	SEC	AUTHORS	TITLE	STATUS
28071	02-19	ART	Dwicahyani, Aldisa	PEMETAAN PREVALENSI DAN ANALISIS FAKTOR PENYAKIT...	Awaiting assignment

Hasil Turnitin

Skripsi_Fadhila Rahma

ORIGINALITY REPORT

19%	17%	12%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	ejournal.bsi.ac.id Internet Source	2%
2	e-journal.my.id Internet Source	1%
3	journal.thamrin.ac.id Internet Source	1%
4	Submitted to STT PLN Student Paper	1%
5	doaj.org Internet Source	1%
6	core.ac.uk Internet Source	1%
7	journal.umkendari.ac.id Internet Source	<1%
8	ejurnal.seminar-id.com Internet Source	<1%

Code Program

```
# Data Preparation
# Import Library
import pandas as pd
import numpy as np
from  sklearn.preprocessing  import  LabelEncoder,  MinMaxScaler,
StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from        sklearn.metrics          import        mean_absolute_error,
mean_squared_error, r2_score, make_scorer
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.inspection import permutation_importance
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
from sklearn import set_config
set_config(display="text")  # Nonaktifkan representasi HTML

# Load Dataset
df = pd.read_csv('/content/TBC_Dataset - TBC.csv')
df

# Columns and Rows
shape = df.shape
print("Jumlah Baris: ", shape[0])
print("Jumlah Kolom: ", shape[1])

# Data Type
df.dtypes

# Data Preprocessing
## Data Cleaning
### Duplicate
# Duplicate
duplicate_count = df.duplicated().sum()
if duplicate_count > 0:
```

```
    print(f"Terdapat {duplicate_count} baris duplikat.")
else:
    print("Tidak ada data duplikat.")


# Drop Duplicate
df = df.drop_duplicates()

# Duplicate
duplicate_count = df.duplicated().sum()
if duplicate_count > 0:
    print(f"Terdapat {duplicate_count} baris duplikat.")
else:
    print("Tidak ada data duplikat.")

### Missing Value
#### Checking Missing Value
# Total Missing Value
missing_values = df.isnull().sum()
print("\nJumlah Missing Values per kolom:")
print(missing_values)

# Proportion Missing Value
proportion = round((df.isnull().sum() / df.shape[0]) * 100, 2)
print("\nJumlah Proporsi Missing Values per kolom:")
print(proportion)

#### Imputation Missing Value
# Imputation Numerical Columns Using Median
numerical_columns = ['Prevalensi TBC', 'Populasi', 'Faktor
Lingkungan','Akses Layanan Kesehatan', 'Kepadatan Penduduk']
for col in numerical_columns:
    df[col].fillna(df[col].median(), inplace=True)

# Imputation Categorical Columns Using Mode
df['Kualitas Udara'].fillna(df['Kualitas Udara'].mode()[0],
inplace=True)

# Missing Value
missing_values = df.isnull().sum()
if missing_values.sum() > 0:
```

```
    print("Kolom dengan missing values:")
    print(missing_values[missing_values > 0])
else:
    print("Tidak ada missing values dalam dataset.")

### Outliers

#### Boxplot
df.info()

# Boxplot Kepadatan Penduduk
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['Kepadatan Penduduk'])
plt.title('Boxplot Kepadatan Penduduk')
plt.xlabel('Kepadatan Penduduk')
plt.show()

# Boxplot Akses Layanan Kesehatan
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['Akses Layanan Kesehatan'])
plt.title('Boxplot Akses Layanan Kesehatan')
plt.xlabel('Akses Layanan Kesehatan')
plt.show()

# Boxplot Faktor Lingkungan
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['Faktor Lingkungan'])
plt.title('Boxplot Faktor Lingkungan')
plt.xlabel('Faktor Lingkungan')
plt.show()

# Boxplot Populasi
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['Populasi'])
plt.title('Boxplot Populasi')
plt.xlabel('Populasi')
plt.show()

# Boxplot Prevalensi TBC
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['Prevalensi TBC'])
```

```
plt.title('Boxplot Prevalensi TBC')
plt.xlabel('Prevalensi TBC')
plt.show()

## Exploratory Data Analysis (EDA)
# Distribusi Data (Histogram)
numerical_columns = ['Prevalensi TBC', 'Populasi', 'Faktor
Lingkungan','Akses Layanan Kesehatan', 'Kepadatan Penduduk']
for col in numerical_columns:
    plt.figure(figsize=(8, 5))
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f'Distribusi {col}')
    plt.xlabel(col)
    plt.ylabel('Frekuensi')
    plt.show()

# Heatmap Korelasi
plt.figure(figsize=(10, 8))
sns.heatmap(df[numerical_columns].corr(), annot=True,
cmap='coolwarm', fmt='.2f')
plt.title('Heatmap Korelasi Antar Variabel')
plt.show()

# Distribusi Kategori (Kualitas Udara)
plt.figure(figsize=(8, 5))
sns.countplot(x='Kualitas Udara', data=df)
plt.title('Distribusi Kategori Kualitas Udara')
plt.xlabel('Kualitas Udara')
plt.ylabel('Jumlah')
plt.show()

# Tren rata-rata Prevalensi TBC per Tahun
avg_tbc_per_year = df.groupby('Tahun')['Prevalensi TBC'].mean()
plt.figure(figsize=(8, 5))
sns.lineplot(x=avg_tbc_per_year.index, y=avg_tbc_per_year.values,
marker='o')
plt.title('Rata-rata Prevalensi TBC per Tahun')
plt.xlabel('Tahun')
plt.ylabel('Rata-rata Prevalensi TBC')
plt.grid()
plt.show()
```

```python
# Distribusi Kualitas Udara per Tahun
plt.figure(figsize=(10, 6))
sns.countplot(x='Tahun', hue='Kualitas Udara', data=df)
plt.title('Distribusi Kualitas Udara per Tahun')
plt.xlabel('Tahun')
plt.ylabel('Jumlah Wilayah')
plt.legend(title='Kualitas Udara', labels=['Baik', 'Sedang',
'Buruk'])
plt.show()

## Data Transformation

### Label Encoding
# Convert Kualitas Udara to LabelEncoding
label_encoder = LabelEncoder()

df["Kualitas Udara"] = label_encoder.fit_transform(df["Kualitas
Udara"])
kualitasUdara_mapping = dict(zip(label_encoder.classes_,
label_encoder.transform(label_encoder.classes_)))
print("\nMapping Kualitas Udara:")
for kategori, kode in kualitasUdara_mapping.items():
    print(f"{kategori} -> {kode}")

## Feature Selection
# Definisikan kolom fitur
features = df[['Populasi', 'Faktor Lingkungan', 'Akses Layanan
Kesehatan',
                  'Kepadatan Penduduk', 'Kualitas Udara']]
features = features.apply(pd.to_numeric, errors='coerce')

features

# Modeling
## K-Means Clustering
# Elbow Method untuk menentukan jumlah cluster optimal
inertia = []
range_clusters = range(1, 11)  # Uji jumlah cluster dari 1 hingga
10
```

```python
for k in range_clusters:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(features)
    inertia.append(kmeans.inertia_)

# Visualisasi Elbow Method
plt.figure(figsize=(8, 5))
plt.plot(range_clusters, inertia, marker='o')
plt.title('Elbow Method untuk Menentukan Jumlah Cluster Optimal')
plt.xlabel('Jumlah Cluster')
plt.ylabel('Inertia')
plt.grid()
plt.show()

# Uji jumlah cluster dari 2 hingga 10
for n_clusters in range(2, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    labels = kmeans.fit_predict(features)
    score = silhouette_score(features, labels)
    print(f"Silhouette Score untuk {n_clusters} cluster:
{score:.4f}")

# Optimal Cluster
optimal_k = 3
kmeans = KMeans(n_clusters=optimal_k, random_state=42)

# Fit K-Means pada seluruh dataset
clusters = kmeans.fit_predict(features) # Use 'features' directly

# Tambahkan hasil clustering sebagai fitur baru
df['Cluster'] = clusters
df.head()

### Model Evaluation
sil_score = silhouette_score(features, clusters)
print(f"Silhouette Score untuk {optimal_k} cluster:
{sil_score:.4f}")

### K-Means Visualization
# Visualisasi hasil clustering
plt.figure(figsize=(8, 6))
```

```
sns.scatterplot(data=df, x='Prevalensi TBC', y='Faktor
Lingkungan',
                hue='Cluster', palette='Set1', s=100)
plt.title('Hasil Clustering: Prevalensi TBC vs Faktor
Lingkungan')
plt.xlabel('Prevalensi TBC')
plt.ylabel('Faktor Lingkungan (%)')
plt.legend(title='Cluster')
plt.grid()
plt.show()

# Populasi vs Kepadatan Penduduk
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Populasi', y='Kepadatan Penduduk',
                hue='Cluster', palette='Set2', s=100)
plt.title('Hasil Clustering: Populasi vs Kepadatan Penduduk')
plt.xlabel('Populasi')
plt.ylabel('Kepadatan Penduduk')
plt.legend(title='Cluster')
plt.grid()
plt.show()

# Cluster Distribution using Bar Plot
cluster_counts = df['Cluster'].value_counts().sort_index()
plt.figure(figsize=(8, 6))
sns.barplot(x=cluster_counts.index, y=cluster_counts.values,
palette='coolwarm')
plt.title('Jumlah Wilayah per Cluster')
plt.xlabel('Cluster')
plt.ylabel('Jumlah Wilayah')
plt.grid()
plt.show()

## Split Data
# Fitur (X) dan Target (y)
X = df[['Populasi', 'Faktor Lingkungan', 'Akses Layanan
Kesehatan', 'Kepadatan Penduduk', 'Kualitas Udara']]
y = df['Prevalensi TBC']

X
```

```
y

# Split data menjadi data latih dan uji
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

print("Shape data latih:", X_train.shape, y_train.shape)
print("Shape data uji:", X_test.shape, y_test.shape)

## Support Vector Regressor
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler

# Standarisasi data (SVR membutuhkan scaling)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Inisialisasi SVR dengan kernel RBF
svr_regressor = SVR(kernel='rbf', C=1.0, epsilon=0.1)

# Latih model
svr_regressor.fit(X_train_scaled, y_train)

# Prediksi pada data uji
y_pred_svr = svr_regressor.predict(X_test_scaled)

# Evaluasi Model
mae_svr = mean_absolute_error(y_test, y_pred_svr)
mse_svr = mean_squared_error(y_test, y_pred_svr)
rmse_svr = np.sqrt(mse_svr)
r2_svr = r2_score(y_test, y_pred_svr)

print(f"Support Vector Regressor:")
print(f"Mean Absolute Error (MAE): {mae_svr:.2f}")
print(f"Mean Squared Error (MSE): {mse_svr:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse_svr:.2f}")
print(f"R² Score: {r2_svr:.2f}")

## Support Vector Regressor Visualization
from sklearn.inspection import permutation_importance
```

```
# Hitung Permutation Feature Importance
perm_importance = permutation_importance(
    svr_regressor, X_test_scaled, y_test,
scoring='neg_mean_absolute_error', random_state=42
)

# Simpan hasil dalam DataFrame
feature_names = X_train.columns  # Ambil nama fitur dari dataset
asli
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': perm_importance.importances_mean
}).sort_values(by='Importance', ascending=False)

# Tampilkan hasil
print(importance_df)

# Visualisasi
plt.figure(figsize=(8, 5))
plt.barh(importance_df['Feature'], importance_df['Importance'])
plt.xlabel('Mean Permutation Importance')
plt.title('Permutation Feature Importance (SVR)')
plt.gca().invert_yaxis()
plt.grid()
plt.show()

# Perbandingan Algoritma Regressor
# Import library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.svm import SVR
```

```python
import xgboost as xgb

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Standarisasi untuk model yang butuh scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Inisialisasi model
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(random_state=42),
    "Random Forest": RandomForestRegressor(n_estimators=100,
random_state=42),
    "Gradient Boosting":
GradientBoostingRegressor(n_estimators=100, random_state=42),
    "XGBoost": xgb.XGBRegressor(n_estimators=100,
random_state=42),
    "SVR": SVR(kernel='rbf')
}

# Evaluasi model
results = {}

for name, model in models.items():
    # Gunakan data yang sudah di-scaling untuk SVR
    if name == "SVR":
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)
    else:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

    # Hitung metrik evaluasi
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)
```

```
    # Simpan hasil
    results[name] = [mae, mse, rmse, r2]

# Buat DataFrame hasil evaluasi
eval_df = pd.DataFrame(results, index=["MAE", "MSE", "RMSE", "R²
Score"]).T
print(eval_df)

# Visualisasi hasil dengan bar chart
eval_df.plot(kind='bar', figsize=(10, 6))
plt.title("Perbandingan Evaluasi Model Regresi")
plt.xticks(rotation=45)
plt.ylabel("Score")
plt.grid()
plt.show()
```