DAFTAR PUSTAKA

Abd. Razak, F., Shitan, M., H. Hashim, A., dkk. (2022). Load Forecasting Using *Time Series* Models. Jurnal Kejuruteraan 21 (2009): 53-62. https://doi.org/10.17576/JKUKM-2009-21-06

Amazon EC2 Auto *Scaling*: Amazon Web Service. (2022). Dipetik Maret 15, 2022, dari aws.amazon: https://aws.amazon.com/blogs/compute/fleet management-made-easy-with-auto-*scaling*/

Addison W., Vernon J., Nicholas R., dkk. (2020). The Autoregressive Linear Mixture Model: A Time-Series Model for an Instantaneous Mixture of Network Processes. IEEE Trans. Signal Process. 68: 4481-4496.

Cerqueira, V., Torgo, L., Mozetic, I. (2020). Evaluating *time series* forecasting models: an empirical study on performance estimation methods. https://doi.org/10.1007/s10994-020-05910-7

Huda, N. (2021). Analisis Kinerja *Website* Dinas Komunikasi dan Informatika Menggunakan Metode Pieces. 10, 155–161.

Lingling, N, dkk. (2020). *Streamflow forecasting using extreme gradient boosting model coupled with Gaussian mixture model*. Journal of Hydrology. https://doi.org/10.1016/j.jhydrol.2020.124901

Mangayarkarasi, M., Tami, s., Kappuchamy, R., dkk. (2021). Highly scalable and load balanced web server on AWS *cloud*. IOP Conf. Ser.: Mater. Sci. Eng. 1055 012113.

Minkenberg, C. (2019). Load Balancing. In Encyclopedia of Computer Graphics and Games (pp. 1-5). Springer.

Naqash A., Yazeed G., Adnan M. (2022).: Load Forecasting Techniques for Power System: Research Challenges and Survey. IEEE Access 10: 71054-71090.

Petropoulos, F., Apiletti, D., Assimakopoulos, V., dkk. (2022). Forecasting: theory and practice. International Journal of Forecasting 38:705–871.

Rajendra, P. (2021). Forecasting Product Order Demand Using SARIMA Model. international Journal for Research in Applied Science & Engineering Technology (IJRAS), Volume 9 Issue X Oct 2021.

Rakhmawati, N., dkk. (2017). A performance evaluation for assessing registered *website*s. Jurnal Kajian Ilmiah. https://pdf.sciencedirectassets.com/280203

.. (2020). Multiple Linear Regression. Principles of Managerial Statistics ıd Data Science, 473–517. doi:10.1002/9781119486473.ch14

Rouse, M. (2019). *Load Balancer*. Retrieved from https://searchnetworking.techtarget.com/definition/load-balancer.

Sasmito, W. S. (2020). Anual Performance Planning System With *Enterprise* Architecture Modelling The Secretariat Of The Central Java Province Parliament Used Framework TOGAF. International Journal of Science and Humanity, 3(4).

Suliman. (2020). Analisis Performa Websitw Universitas Teuku Umar dan Universitas Samudera Menggunakan Pingdom Tools dan Gtmetrix. SIMKOM, Vol. 5, No. 1. http://ejurnal.stmikbinsa.ac.id/index.php/simkom

Thoyyib, A, IGL. Eka, P. (2022). Analisis Performansi Web Server Menggunakan Load Balancing Pada Virtualisasi Docker Container. JINA CS: Volume 03 Nomor 04, 2022 (Journal of Informatics and Computer Science) ISSN: 2686-2220.

Ulgen, T., Poyrazoglu, G. (2020). *Predictor Analysis for Electricity Price Forecasting by Multiple Linear Regression. International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM).* doi:10.1109/speedam48782.2020.9161866

Wang, C. (2022). *Machine Learning Prediction of Turning Precision Using Optimized XGBoost Model*. Appl. Sci. 2022, 12, 7739. https://doi.org/ 10.3390/app12157739

What is web server: Developer Mozilla. (2022). Dipetik Maret 15, 2022, dari developer.mozilla.org: https://developer.mozilla.org/enUS/docs/Learn/Common_questions/What_is_a_web_server

Yoganingrum, A., Sensuse, D. I., Murni, A. (2022). A Taxonomy of *Enterprise* Architecture Framework for Indonesian SMEs. International Journal of Computer Science Issue (IJCSI), 10(2).

Lampiran 1 *Source Code Import* Data

```
import numpy as np
import pandas as pd
import pandas_profiling
import warnings
warnings.filterwarnings('ignore')
import datetime
from datetime import date
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set_style("whitegrid")
import cufflinks as cf
import plotly.express as px
from plotly.offline import download_plotlyjs, init_notebook_mode,
plot, iplot
init_notebook_mode(connected=True)
cf.go_offline()
import pandas_profiling
import plotly.graph_objects as go
from sklearn.model_selection import train_test_split,
cross_val_score, GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
import xgboost as xg


df=pd.read_csv('daily-website-visitors.csv')
df.rename(columns = {'Day.Of.Week':'day_of_week'
                     ,'Page.Loads':'page_loads'
                     ,'Unique.Visits':'unique_visits'
                     ,'First.Time.Visits':'first_visits'
                     ,'Returning.Visits':'returning_visits'},
                    inplace = True)
df=df.replace(',','',regex=True)
df['page_loads']=df['page_loads'].astype(int)
df['unique_visits']=df['unique_visits'].astype(int)
df['first_visits']=df['first_visits'].astype(int)
df['returning_visits']=df['returning_visits'].astype(int)
df
```

Lampiran 2 Filter Data Tahun 2020

```
# Filter data for year 2020
df_2020 = df[df['year'] == 2020]

# Create line chart for returning visits in 2020
fig = px.line(df_2020, x='Date', y='unique_visits',
title='Returning Visits in 2020')

# Optional customization
fig.update_traces(marker_line_color='darkblue',
marker_line_width=2)  # Adjust line color and width
fig.show()
```

Lampiran 3 Dekomposisi Musiman Aditif untuk Data Kunjungan *Website*

```
import statsmodels.api as sm

# Perform Additive Seasonal Decomposition
decomposition = sm.tsa.seasonal_decompose(y_2020,
model='additive', period=7)  # Assuming weekly seasonality

# Plot the decomposed components
plt.figure(figsize=(16, 8))
plt.subplot(411)
plt.plot(decomposition.trend, label='Trend')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(decomposition.seasonal, label='Seasonality')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(decomposition.resid, label='Residuals')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(y_2020, label='Original')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

Lampiran 4 Analisis Autokorelasi dan Autokorelasi Parsial untuk Prediksi Kunjungan *Website*

```
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Filter data for the year 2020
df_2020 = df[df['year'] == 2020]

# Create predictor variable 'days_f'
df_2020['days_f'] = np.where((df_2020['Day'] == 'Monday') |
                             (df_2020['Day'] == 'Tuesday') |
                             (df_2020['Day'] == 'Wednesday') |
                             (df_2020['Day'] == 'Thursday') |
                             (df_2020['Day'] == 'Friday') |
                             (df_2020['Day'] == 'Saturday') |
                             (df_2020['Day'] == 'Sunday'), 1, 0)

# Drop the 'Day' column as it's no longer needed
df_2020.drop('Day', axis=1, inplace=True)

# Prepare predictors and target variables
X_2020 = df_2020[['page_loads', 'first_visits',
'returning_visits', 'days_f']]
y_2020 = df_2020['unique_visits']

# Split the data into train and test sets
X_train_2020, X_test_2020, y_train_2020, y_test_2020 =
train_test_split(X_2020, y_2020, test_size=0.3, random_state=42)

# Train the linear regression model
regressor_2020 = LinearRegression(fit_intercept=False,
normalize=True)
regressor_2020.fit(X_train_2020, y_train_2020)

# Predict unique visits for 2020
y_pred_2020 = regressor_2020.predict(X_test_2020)

# Create DataFrame to store actual and predicted values
lr_2020 = pd.DataFrame({'Actual': y_test_2020, 'Predicted':
y_pred_2020})

# Autocorrelation plot
fig, ax = plt.subplots(figsize=(12, 6))
    f(y_test_2020, ax=ax, lags=40)
    le('Autocorrelation Plot')
    bel('Lag')
    bel('Autocorrelation')
    w()
```

```
# Partial autocorrelation plot
fig, ax = plt.subplots(figsize=(12, 6))
plot_pacf(y_test_2020, ax=ax, lags=35)  # Adjust lags to a value
less than or equal to 35
plt.title('Partial Autocorrelation Plot')
plt.xlabel('Lag')
plt.ylabel('Partial Autocorrelation')
plt.show()
```

Lampiran 5 Prediksi Jumlah Kunjungan *Website* Menggunakan *Multi Linear Regression* Model

```
# Filter data for the year 2020
df_2020 = df[df['year'] == 2020]

# Create predictor variable 'days_f'
df_2020['days_f'] = np.where((df_2020['Day'] == 'Monday') |
                             (df_2020['Day'] == 'Tuesday') |
                             (df_2020['Day'] == 'Wednesday') |
                             (df_2020['Day'] == 'Thursday') |
                             (df_2020['Day'] == 'Friday') |
                             (df_2020['Day'] == 'Saturday') |
                             (df_2020['Day'] == 'Sunday'), 1, 0)

# Drop the 'Day' column as it's no longer needed
df_2020.drop('Day', axis=1, inplace=True)

# Prepare predictors and target variables
X_2020 = df_2020[['page_loads', 'first_visits',
'returning_visits', 'days_f']]
y_2020 = df_2020['unique_visits']

# Introduce noise into the target variable
noise = np.random.normal(0, 10, y_2020.shape)
# Adjust noise standard deviation
y_2020_noisy = y_2020 + noise

# Split the data into train and test sets
X_train_2020, X_test_2020, y_train_2020, y_test_2020 =
train_test_split(X_2020, y_2020, test_size=0.3, random_state=42)

# Train the linear regression model
regressor_2020 = LinearRegression(fit_intercept=False,
    ze=True)
    or_2020.fit(X_train_2020, y_train_2020)

    ct unique visits for 2020
    2020 = regressor_2020.predict(X_test_2020)
```

```
# Create DataFrame to store actual and predicted values
lr_2020 = pd.DataFrame({'Actual': y_test_2020, 'Predicted':
y_pred_2020})

# Plot the actual vs predicted values
plt.figure(figsize=(16, 8))
sns.lineplot(data=lr_2020)

# Customize plot if needed
plt.title('Forecast Using Multi Linear Regression Model for 2020')
plt.xlabel('Time')
plt.ylabel('Unique Visits')
plt.grid(True)
plt.legend(loc='best')
plt.show()
```

Lampiran 6 Prediksi Jumlah Kunjungan *Website* Menggunakan *XGBoost Regression* Model

```
# Filter data for the year 2020
df_2020 = df[df['year'] == 2020]

# Create predictor variable 'days_f'
df_2020['days_f'] = np.where((df_2020['Day'] == 'Monday') |
                             (df_2020['Day'] == 'Tuesday') |
                             (df_2020['Day'] == 'Wednesday') |
                             (df_2020['Day'] == 'Thursday') |
                             (df_2020['Day'] == 'Friday') |
                             (df_2020['Day'] == 'Saturday') |
                             (df_2020['Day'] == 'Sunday'), 1, 0)

# Drop the 'Day' column as it's no longer needed
df_2020.drop('Day', axis=1, inplace=True)

# Prepare predictors and target variables
X2_2020 = df_2020[['page_loads', 'first_visits',
'returning_visits', 'days_f']]
y2_2020 = df_2020['unique_visits']

# Split the data into train and test sets
X_train_2020, X_test_2020, y_train_2020, y_test_2020 =
train_test_split(X2_2020, y2_2020, test_size=0.3, random_state=42)

    the XGBoost regressor
    xg.XGBRegressor(objective='reg:squarederror',
ators=10, seed=123)
it(X_train_2020, y_train_2020)

ct unique visits for 2020
```

```
xgb_pred_2020 = xgb_r.predict(X_test_2020)

# Create DataFrame to store actual and predicted values
xgb_df_2020 = pd.DataFrame({'Actual': y_test_2020, 'Predicted':
xgb_pred_2020})
# Plot the actual vs predicted values
plt.figure(figsize=(16, 8))
sns.lineplot(data=xgb_df_2020)

# Customize plot if needed
plt.title('Forecast Using XGBoost Regression Model for 2020')
plt.xlabel('Time')
plt.ylabel('Unique Visits')
plt.grid(True)
plt.legend(loc='best')
```

Lampiran 7 Evaluasi Kinerja *Multi Linear Regression* Model

```
#Metode 1
regressor2.score(X_test,y_test)*100

#Metode 2
from sklearn.metrics import mean_absolute_error,
mean_squared_error, mean_absolute_percentage_error
mae = mean_absolute_error(y_test, y_pred2)
mse = mean_squared_error(y_test, y_pred2)
rmse = np.sqrt(mse)  # Root Mean Squared Error
mape = mean_absolute_percentage_error(y_test, y_pred2) * 100  #
MAPE in percentage
print(f"MAE: {mae:.2f}")
print(f"MSE: {mse:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"MAPE: {mape:.2f}%")
```

Lampiran 8 Evaluasi Kinerja *XGBoost* Model

```
#Metode 1
xgb_r.score(X_test,y_test)*100

#Metode 2
from sklearn.metrics import mean_absolute_error,
mean_squared_error, mean_absolute_percentage_error

ean_absolute_error(y_test, xgb_pred)
ean_squared_error(y_test, xgb_pred)
np.sqrt(mse)  # Root Mean Squared Error
mean_absolute_percentage_error(y_test, xgb_pred) * 100  #
 percentage
```

```
print(f"MAE: {mae:.2f}")
print(f"MSE: {mse:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"MAPE: {mape:.2f}%")
```

Lampiran 9 *Schedule Automatic Jobs* (*Simulation*)

```
package main

import (
    "encoding/csv" // Package for reading and writing CSV files
    "fmt"          // Package for formatted I/O
    "io/ioutil"    // Package for I/O utility functions
    "net/http"// Package for HTTP client and server implementation
    "os"           // Package for OS functions
    "os/exec"      // Package for executing external commands
    "strconv"      // Package for string conversions
    "time"         // Package for time functions
    "github.com/robfig/cron/v3" // Package for job scheduling
)

func main() {
    // Define HTTP handlers for various endpoints
    http.HandleFunc("/consume-memory/", consumeMemoryHandler)
    http.HandleFunc("/status/200", status200Handler)
    http.HandleFunc("/status/400", status400Handler)
    http.HandleFunc("/status/500", status500Handler)

    // Create a new cron scheduler with a fixed timezone
    scheduler :=
    cron.New(cron.WithLocation(time.FixedZone("Asia/Jakarta",
    7*60*60)))

    // Schedule jobs and list them
    Job(scheduler)
    ListJobs(scheduler)

    // Start HTTP server
    fmt.Println("Server listening on port 8080...")
    http.ListenAndServe(":8080", nil)
}

func status200Handler(w http.ResponseWriter, r *http.Request) {
    riteHeader(http.StatusOK)

    Get IP address from an external service
     err := http.Get("http://checkip.amazonaws.com")
    err != nil {
     fmt.Println("Failed to get IP:", err)
```

```
        return
    }
    defer ip.Body.Close()

    ipAddress, err := ioutil.ReadAll(ip.Body)
    if err != nil {
        fmt.Println("Failed to read IP:", err)
        return
    }

    fmt.Println("IP Address:", string(ipAddress))
    fmt.Fprintf(w, "Status 200 - OK\n"+"IP Address:
    "+string(ipAddress)+"\n")
}

func status400Handler(w http.ResponseWriter, r *http.Request) {
    w.WriteHeader(http.StatusBadRequest)
    fmt.Fprintf(w, "Status 400 - Bad Request\n")}

func status500Handler(w http.ResponseWriter, r *http.Request) {
    w.WriteHeader(http.StatusInternalServerError)
    fmt.Fprintf(w, "Status 500 - Internal Server Error\n")}

func consumeMemoryHandler(w http.ResponseWriter, r *http.Request){
    // Extract the number of megabytes from the URL path
    mbStr := r.URL.Path[len("/consume-memory/"):]
    mb, err := strconv.Atoi(mbStr)
    if err != nil {
        http.Error(w, "Invalid number of megabytes",
        http.StatusBadRequest)
        return
    }

    // Allocate memory
    data := make([]byte, mb*1024*1024)
    if data == nil {
        http.Error(w, "Failed to allocate memory",
        http.StatusInternalServerError)
        return
    }

    fmt.Fprintf(w, "Allocated %d MB of memory", mb)
}

    b(scheduler *cron.Cron) {
    Open CSV file
    e, err := os.Open("data.csv")
    err != nil {
        fmt.Println("Failed to open file:", err)
```

```
        return
    }
    defer file.Close()

    // Read CSV file
    reader := csv.NewReader(file)
    records, err := reader.ReadAll()
    if err != nil {
        fmt.Println("Failed to read file:", err)
        return
    }

    for i, record := range records {
        // Skip header row
        if i == 0 {
            continue
        }

        // Parse datetime and number of requests from CSV record
        dateTimeStr := record[0] + " " + record[1]
        requests := record[2]

        dateTime, err := time.Parse("2/1/2006 15:04:05",
        dateTimeStr)
        if err != nil {
            fmt.Println("Failed to parse date time:", err)
            return
        }

        // Create cron expression based on the parsed datetime
        cronExpr := fmt.Sprintf("%d %d %d %d *",
        dateTime.Minute(), dateTime.Hour(), dateTime.Day(),
        int(dateTime.Month()))
        scheduler.AddFunc(cronExpr, func(req string) func() {
            return func() {
                RunApacheBenchmark(req, "1000")
            }
        }(requests))
    }

    // Start the scheduler
    scheduler.Start()
}

nApacheBenchmark(requests string, concurrency string) {
Execute Apache Benchmark with the specified number of
requests and concurrency
```

```
    cmd := exec.Command("ab", "-n", requests, "-c", concurrency,
           "http://simulationloadwebtest-473289163.ap-southeast-
           2.elb.amazonaws.com/status/200")
    if output, err := cmd.CombinedOutput(); err != nil {
        fmt.Printf("Failed to run Apache Benchmark: %s\nError:
        %s\n", err, output)
    } else {
        fmt.Printf("Apache Benchmark ran successfully: %s\n",
        output)
    }
}

func ListJobs(scheduler *cron.Cron) {
    // List all scheduled jobs
    entries := scheduler.Entries()
    fmt.Println("Scheduled Jobs:")
    for _, entry := range entries {
        fmt.Printf("ID: %d, Schedule: %s, Next: %s, Prev: %s\n",
            entry.ID, entry.Schedule, entry.Next, entry.Prev)
    }
}
```

Lampiran 10 Lembar Perbaikan Skripsi

LEMBAR PERBAIKAN SKRIPSI

"ANALISIS PERFORMA *WEBSITE* DENGAN MENGGUNAKAN *FORECAST INSTANCE LOAD BALANCER* SKALA *ENTERPRISE* DENGAN MODEL *MULTI LINEAR REGRESSION* DAN *XGBOOST REGRESSION*"

OLEH:

NUR ISLAMIAH RIFAI
D121201006

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 19 Juli 2024.

Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	Tanda Tangan
Ketua	Dr. Eng. Ir. Zulkifli Tahir, S.T., M.SC.	
Sekretaris	Ir. Muhammad Alief Fahdal Imran Oemar, S.T., M.Sc.	
Anggota	A. Ais Prayogi Alimuddin, S.T., M.Eng.	
	Prof. Dr. Ir. Andani, M.T.	

Persetujuan Perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Dr. Eng. Ir. Zulkifli Tahir, S.T., M.SC.	
II	Ir. Muhammad Alief Fahdal Imran Oemar, S.T., M.Sc.	