DAFTAR PUSTAKA

Abdillah. (2013). *Progam Linear.* Makassar: Dua Satu Press.

Amril, M., Tukino, Sutan, F., Ahmad, F., & Ilman, K. (2020). Klasifikasi Untuk Prediksi Cuaca Menggunakan Esemble Learning. *Jurnal Pengkajian dan Penerapan Teknik Informatika*, 138-146.

Azmi, Z., & Dahria, M. (2013). Decision Tree Berbasis Algoritma untuk Pengambilan Keputusan. *Jurnal Ilmiah Saintikom*, 157-164.

Dynamic Education Group. (2020, Februari 13). *USA*. Retrieved from dynamicedugroup.com: https://www.dynamicedugroup.com/en/usa/

Guivera. (2020, Maret 23). *Aritkel Kegiatan Waktu Kerja*. Retrieved from Time Management Guide: https://www.time-management-guide.com/gaya-waktu-kerja-ala-amerika-kayak-gimana-itu/

Guivera. (2020, Maret 23). *Gaya Waktu Kerja Ala Amerika Kayak Gimana Itu?* Retrieved from Time Management Guide: https://www.time-management-guide.com/gaya-waktu-kerja-ala-amerika-kayak-gimana-itu/

Harmon, C., Iwan, M., Tintin, S., & Rahil, J. (2019). Faktor Kunci Keberhasilan Ritel Modern Di Indonesia. *Jurnal Akutansi, Ekonomi dan Manajemen Bisnis*, 201-208.

Lopatin, J., Dolos, K., Hernandez, J., Galleguillos, M., & Fassnacht F, E. (2016). Comparing Generalized Linear Models and random forest to model vascular . *Remote Sensing of Environment*, 1-26.

J.J. Faraway. (2010). *Generalized Linear Models,*. Bath: Elsevier.

Jong, P. D., & Heller, G. Z. (2008). *Models For Insurance Data.* New York: Cambridge University Press.

K. V. Mardia, J. T. (1997). *Multivariate Analysis.* Boston: Academic Press.

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling.* New York: Springer.

Louppe, G. (2014). *Understanding Random Forest.* Liège.

Nelder, J., & Wedderburn, R. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society. Series A(General)*, 370-384.

Supriyadi, E., Mariani, S., & Sugiman. (2017). Perbandingan Metode Partial Least Square(PLS) dan Principal Component Regression(PCR) untuk Mengatasi

Multikolinearitas pada Model Regresi Linear Berganda. *UNNES Journal of Mathamatics.*

Zahro, J., Caraka, R. E., & Herliansyah, R. (2018). *Aplikasi Generalized Linear Model Pada R.* Yogyakarta: Innosain.

LAMPIRAN

1. Data Pejalan Kaki di Jembatan Brooklyn

   https://drive.google.com/drive/folders/1aRIAMRwLEM4C2R1R7iDz8SuxaoJ21pCV?usp=sharing

2. Coding Decision Tree

   - Hour_1

```
library(rpart)
library(rpart.plot)
olah<-Hour_1[,c(1,2,4,8,9,12,18)]
str(olah)
set.seed(123)
train<-olah
m1 <- rpart(Pedestrians~., data=train,method="anova")
rpart.plot(m1)
summary(m1)
```

   - Hour_2

```
library(rpart)
library(rpart.plot)
olah<-Hour_2[,c(1,3,6,7,8,11)]
str(olah)
set.seed(123)
train<-olah
m1 <- rpart(Pedestrians~., data=train,method="anova")
rpart.plot(m1)
summary(m1)
```

   - Hour_3

```
library(rpart)
library(rpart.plot)
olah<-Hour_3[,c(1,3,7,8,11,19)]
str(olah)
set.seed(123)
train<-olah
m1 <- rpart(Pedestrians~., data=train,method="anova")
rpart.plot(m1)
summary(m1)
```

- Temperature_1

```
library(rpart)
library(rpart.plot)
olah<-Temperature_1[,c(1,3,7,8,11,13)]
str(olah)
set.seed(123)
train<-olah
m1 <- rpart(Pedestrians~., data=train,method="anova")
rpart.plot(m1)
summary(m1)
```

- Temperature_2

```
library(rpart)
library(rpart.plot)
olah<-Temperature_2[,c(2,4,5,6,9)]
str(olah)
set.seed(123)
train<-olah
m1 <- rpart(Pedestrian~., data=train,method="anova")
rpart.plot(m1)
summary(m1)
```

3. Coding Random Forest

- Hour_1

```
import pandas as pd
import numpy as np
dataset = pd.read_excel("Hour_1.xlsx")
dataset.head()
dataset.isnull().sum()
dataset.dropna(axis = 0, inplace = True)
dataset.isnull().sum()
dataset.head()
dataset.drop(['Location'], axis=1, inplace=True)
dataset.drop(['Towards Manhattan'], axis=1, inplace=True)
dataset.drop(['Towards Brooklyn'], axis=1, inplace=True)
dataset.drop(['Weather_Summary'], axis=1, inplace=True)
dataset.drop(['Lat'], axis=1, inplace=True)
dataset.drop(['Long'], axis=1, inplace=True)
dataset.drop(['Location.1'], axis=1, inplace=True)
```

```
dataset.drop(['Date'], axis=1, inplace=True)
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
#Define dependet variable
Y = dataset['Pedestrians'].values
#Define Independent variable
X = dataset.drop(labels= ['Pedestrians'],axis=1)
#split data into train and test datasets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.4,random_state=20)
from sklearn.ensemble import RandomForestRegressor
model= RandomForestRegressor(n_estimators =150 , random_state=1)
rgs= model.fit(X_train, Y_train)
print (X_test, Y_test)
predict_test = model.predict(X_test)
print(predict_test)
model.score(X_test, Y_test)
from sklearn import tree
rgs.estimators_
# Limit depth of tree to 3 levels
rf_small = RandomForestRegressor(n_estimators=150, max_depth = 3)
rf_small.fit(X_train, Y_train)
plt.figure(figsize= (20,15))
tree.plot_tree(rf_small.estimators_[149],filled=True)
plt.show()
importances = list(rgs.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(X, importances)]
```

```
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances];
```

- Hour_2

```
import pandas as pd
import numpy as np
dataset = pd.read_excel("Hour_2.xlsx")
dataset.head()
dataset.isnull().sum()
dataset.dropna(axis = 0, inplace = True)
dataset.isnull().sum()
dataset.head()
dataset.drop(['Towards Manhattan'], axis=1, inplace=True)
dataset.drop(['Towards Brooklyn'], axis=1, inplace=True)
dataset.drop(['Weather_Summary'], axis=1, inplace=True)
dataset.drop(['Lat'], axis=1, inplace=True)
dataset.drop(['Long'], axis=1, inplace=True)
dataset.drop(['Hour'], axis=1, inplace=True)
dataset.drop(['Location.1'], axis=1, inplace=True)
dataset.drop(['Location'], axis=1, inplace=True)
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
#Define dependet variable
Y = dataset['Pedestrians'].values
#Define Independent variable
X = dataset.drop(labels= ['Pedestrians'],axis=1)
#split data into train and test datasets
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.5,random_state=20)
from sklearn.ensemble import RandomForestRegressor
model= RandomForestRegressor(n_estimators =150 , random_state=1)
rgs= model.fit(X_train, Y_train)
print (X_test, Y_test)
model.score(X_test, Y_test)
from sklearn import tree
rgs.estimators_
# Limit depth of tree to 3 levels
rf_small = RandomForestRegressor(n_estimators = 150, max_depth = 3)
rf_small.fit(X_train, Y_train)
tree_small = rf_small.estimators_[10]
print(tree_small)
plt.figure(figsize = (20,15))
tree.plot_tree(rf_small.estimators_[149],filled = True)
plt.show()
importances = list(rgs.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(X, importances)]
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances];
```

- Hour_3

```
import pandas as pd
import numpy as np
dataset = pd.read_excel("Hour_3.xlsx")
dataset.head()
dataset.isnull().sum()
dataset.dropna(axis = 0, inplace = True)
dataset.isnull().sum()
dataset.head()
dataset.drop(['Towards Manhattan'], axis=1, inplace=True)
dataset.drop(['Towards Brooklyn'], axis=1, inplace=True)
dataset.drop(['Weather_Summary'], axis=1, inplace=True)
dataset.drop(['Lat'], axis=1, inplace=True)
dataset.drop(['Long'], axis=1, inplace=True)
dataset.drop(['Location'], axis=1, inplace=True)
dataset.drop(['Location.1'], axis=1, inplace=True)
dataset.drop(['Hour1'], axis=1, inplace=True)
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
#Define dependet variable
Y = dataset['Pedestrians'].values
#Define Independent variable
X = dataset.drop(labels= ['Pedestrians'],axis=1)
#split data into train and test datasets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.4,random_state = 20)
print (X_test, Y_test)
predict_test = model.predict(X_test)
print(predict_test)
```

```
model.score(X_test, Y_test)
from sklearn import tree
rgs.estimators_
# Limit depth of tree to 3 levels
rf_small = RandomForestRegressor(n_estimators=150, max_depth =3)
rf_small.fit(X_train, Y_train)
tree_small = rf_small.estimators_[10]
print(tree_small)
plt.figure(figsize= (20,15))
tree.plot_tree(rf_small.estimators_[149],filled=True)
plt.show()
importances = list(rgs.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(X, importances)]
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances];
```

- Temperature_1

```
import pandas as pd
import numpy as np
dataset = pd.read_excel("Temperature_1.xlsx")
dataset.head()
dataset.isnull().sum()
dataset.dropna(axis = 0, inplace = True)
dataset.isnull().sum()
dataset.drop(['Towards Manhattan'], axis=1, inplace=True)
dataset.drop(['Towards Brooklyn'], axis=1, inplace=True)
dataset.drop(['Weather_Summary'], axis=1, inplace=True)
dataset.drop(['Lat'], axis=1, inplace=True)
```

```
dataset.drop(['Long'], axis=1, inplace=True)
dataset.drop(['Location'], axis=1, inplace=True)
dataset.drop(['Location.1'], axis=1, inplace=True)
dataset.drop(['Hour1'], axis=1, inplace=True)
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
#Define dependet variable
Y = dataset['Pedestrians'].values
#Define Independent variable
X = dataset.drop(labels=['Pedestrians'],axis=1)
dataset.head()
#split data into train and test datasets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split (X, Y, test_size =
0.4 ,random_state = 20)
from sklearn.ensemble import RandomForestRegressor
model= RandomForestRegressor(n_estimators =150 , random_state=1)
rgs= model.fit(X_train, Y_train)
print (X_test, Y_test)
predict_test = model.predict(X_test)
print(predict_test)
model.score(X_test, Y_test)
from sklearn import tree
rgs.estimators_
# Limit depth of tree to 3 levels
rf_small = RandomForestRegressor(n_estimators= 150, max_depth =  3)
rf_small.fit(X_train, Y_train)
plt.figure(figsize=(20,15))
tree.plot_tree(rf_small.estimators_[149],filled=True)
plt.show()
```

```
importances = list(rgs.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature,
importance in zip(X, importances)]
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in
feature_importances];
```

- Temperature_2

```
import pandas as pd
import numpy as np
dataset = pd.read_excel("Temperature_2.xlsx")
dataset.head()
dataset.isnull().sum()
dataset.dropna(axis = 0, inplace = True)
dataset.isnull().sum()
dataset.drop(['Lat'], axis=1, inplace=True)
dataset.drop(['Long'], axis=1, inplace=True)
dataset.drop(['Location'], axis=1, inplace=True)
dataset.drop(['Location.1'], axis=1, inplace=True)
dataset.drop(['Hour'], axis=1, inplace=True)
dataset.drop(['date'], axis=1, inplace=True)
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
#Define dependet variable
Y = dataset['Pedestrian'].values
#Define Independent variable
X = dataset.drop(labels= ['Pedestrian'],axis=1)
dataset.head()
#split data into train and test datasets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.4,
random_state = 20)
```

```
from sklearn.ensemble import RandomForestRegressor
model=  RandomForestRegressor(n_estimators = 150 , random_state =
1)
rgs=  model.fit(X_train, Y_train)
print (X_test, Y_test)
predict_test = model.predict(X_test)
print(predict_test)
model.score(X_test, Y_test)
from sklearn import tree
rgs.estimators_
# Limit depth of tree to 3 levels
rf_small = RandomForestRegressor(n_estimators = 150, max_depth =
3)
rf_small.fit(X_train, Y_train)
plt.figure(figsize = (20,15))
tree.plot_tree(rf_small.estimators_[149],filled=True)
plt.show()
importances = list(rgs.feature_importances_)
feature_importances = [(feature, round(importance, 2)) for feature,
importance in zip(X, importances)]
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in
feature_importances];
```

4. Coding Generalized Linear Model

- Hour_1

```
lmp <- glm(Pedestrians~ Hour_1 + Weather_new + Temperature_1 + Precipation + Events, family = poisson(link = 'log'),data = Hour_)
summary(lmp)
anova(lmp)
plot(lmp)

library(MASS)
a<- glm(Pedestrians ~ Hour_1 + Weather_new+ Temperature_1 + Precipation + Events, family = negative.binomial(2), data = Hour_)
plot(a)
summary(a)
```

- Hour_2

```
lmp <- glm(Pedestrians~ Hour + Weather_Summary + Temperature + Precipation + Events, family = poisson(link = 'log'),data = Hour_2)
summary(lmp)
anova(lmp)
plot(lmp)

library(MASS)
a<- glm(Pedestrians~ Hour + Weather_Summary + Temperature + Precipation + Events, family = negative.binomial(2), data = Hour_2)
plot(a)
summary(a)
anova(a)
```

- Hour_3

```
lmp <- glm(Pedestrians~ Hour_3 + Weather_new + Temperature_1 + Precipation + Events, family = poisson(link = 'log'),data = Hour_3)
summary(lmp)
anova(lmp)
plot(lmp)

library(MASS)
a<- glm(Pedestrians~ Hour_3 + Weather_new + Temperature_1 + Precipation + Events, family = negative.binomial(2), data = Hour_3)
plot(a)
summary(a)
```

- Temperature_1

```
lmp <- glm(Pedestrians~ Hour + Weather_new + Precipation + Temperature_1 + Event, family = poisson(link = 'log'),data = Temperature_1)
summary(lmp)
anova(lmp)
plot(lmp)


library(MASS)
a<- glm(Pedestrians ~ Hour + Weather_new + Precipation + Temperature_1 + Event, family = negative.binomial(2), data = Temperature_1)
plot(a)
summary(a)
```

- Temperature_2

```
olah<-Temperature_2[,c(1,2,3,6,7,10)]
lmp <- glm(Pedestrian~ Temperature_2 + Precipation + Event, family = poisson(link = 'log'),data = olah)
summary(lmp)
anova(lmp)
plot(lmp)

library(MASS)
a<- glm(Pedestrian~ Temperature_2 + Precipation + Event, family = negative.binomial(2), data = Temperature_2)
plot(a)
summary(a)
```