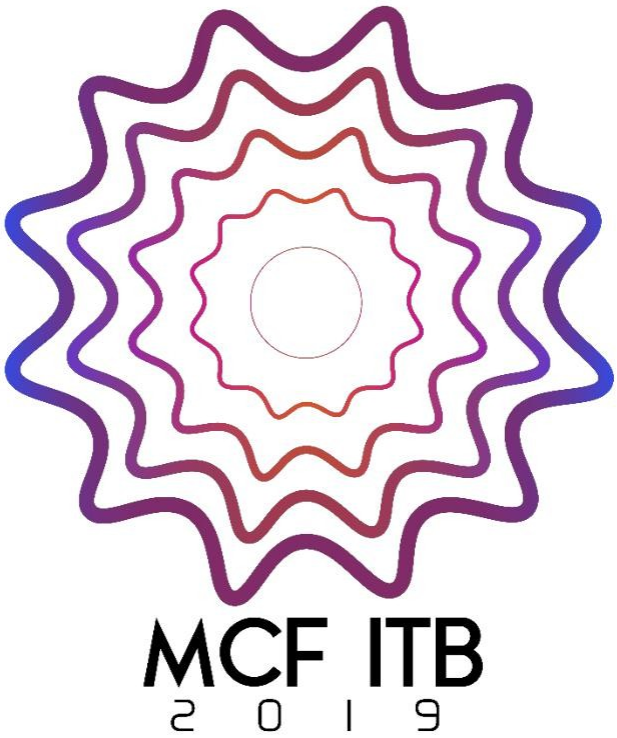
LAPORAN PEMODELAN



-4195-

DSC

KATA PENGANTAR

Alhamdulillah puji dan syukur penulis panjatkan kehadirat Allah SWT. Tuhan Yang Maha Esa dengan segala kuasa-Nya, Dzat yang Maha Pengasih dengan segala kasih saying-Nya, yang terlepas dari segala sifat lemah semua makhluk-Nya. Alhamdulillah, berkat rahmat dan hidayah-Nya, penulis dapat menyelesaikan Laporan Data Science "Bike Share Demand".

Shalawat serta salam mahabbah semoga senantiasa dilimpahkan kepada Nabi Muhammad SAW sebagai pembawa risalah Allah SWT terakhir dan penyempurna seluruh risalah-Nya. dengan segala kerendahan hati izinkanlah penulis untuk menyampaikan terima kasih dan penghargaan yang setinggi-tingginya kepada semua pihak yang telah berjasa memberikan motivasi, bimbingan, serta dukungan dalam rangka menyelesaikan Laporan Data Science "Bike Share Demand".

Semoga kebaikan yang diberikan oleh semua pihak kepada penulis menjadi amal sholeh yang senantiasa mendapat balasan dan kebaikan yang berlipat ganda dari Allah SWT. Aamin.

Akhir kata, penulis menyadari bahwa masih terdapat kekurangan dalam laporan ini, untuk itu saran dan kritik yang sifatnya membangun sangat penulis harapkan.

Depok, 23 Agustus 2019

Team Atom

DAFTAR ISI

ABSTRAK

Bike Share Demand adalah Kompetisi Data Science yang diadakan MCF ITB dengan GOJEK. Permasalahan yang diberikan adalah Bike Sharing System yang merupakan cara menyewa sepeda dimana untuk mendapatkan keanggotaan, penyewaan dan pengembalian sepeda melalui sebuah jaringan kios. Dengan sistem ini, orang dapat menyewa sepeda di satu lokasi dan mengembalikannya di tempat lain. Dari sistem tersebut, ada kemungkinan untuk sepeda bisa penuh atau kosong ketika seorang penyewa datang ke tempat penyewaan. Dengan demikian untuk memprediksi penggunaan sistem seperti itu dapat membantu bagi pengguna untuk merencanakan perjalanan mereka dan juga bagi pelaksana program Capital Bikeshare di Washington, D.C. untuk mengatur sistem dengan benar. Maka di laporan ini menyajikan berbagai cara untuk memprediksi jumlah sepeda yang dapat disewa dalam sistem seperti itu. Prediksi dibuat untuk setiap jam sehari.

Langkah Kerja

A. Data Wrangling

a. Mengekstrak dataset yang disediakan panitia yang berupa dua tabel berformat csv.
b. Melakukan pengecekan semua variabel yang tersedia apakah didalam variabel tersebut terdapat NULL.

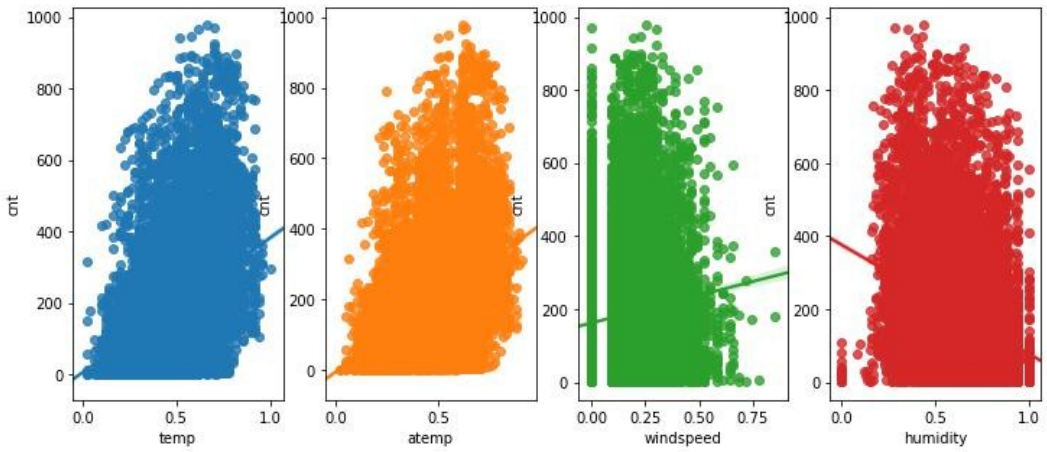
1. Reshaping Data

   Reshaping data yaitu mengubah bentuk data. Kami me-reshaping data dengan :

   a. Membuat kolom baru yaitu kolom weekday, year, dan hour yang berasal dari kolom datetime. Selain itu kami juga membuat grup jam menjadi 6 grup yaitu grup 1 untuk jam lebih dari atau sama dengan 0 dan kurang dari atau sama dengan 6, grup 2 untuk jam sama dengan 7 atau sama dengan 9, grup 3 untuk jam sama dengan 8 atau sama dengan 16 , grup 4 untuk jam lebih dari atau sama dengan 10 dan kurang dari atau sama dengan 15, grup 5 untuk jam sama dengan 17 atau sama dengan 18, dan grup 6 lebih dari atau sama dengan 20.
   b. Mengubah tipe data dari variabel season, weather, holiday, dan workingday pada train dan test dengan type data string
   c. Mengubah variable pada train dan test ke dalam bentuk dummy variables.
   d. Membagi data secara manual dan menhapus kolom yang menurut kami tidak dapat membantu perhitungan prediksi.

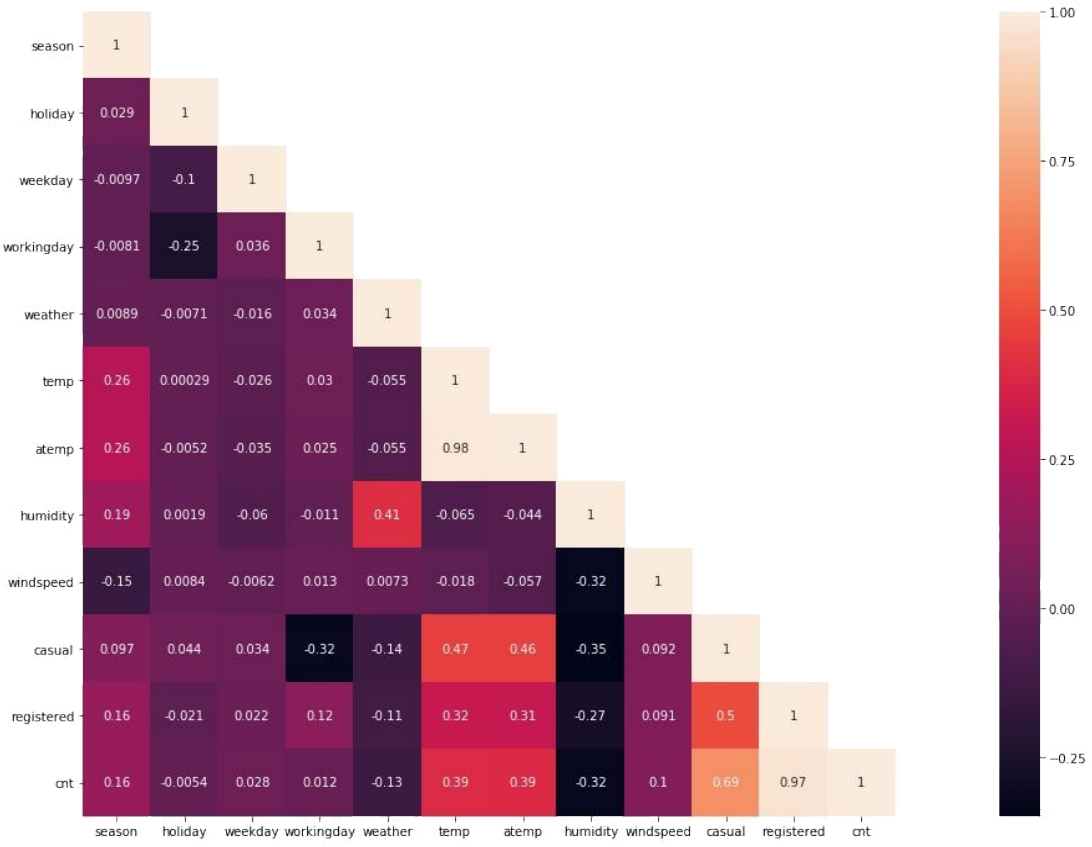
B. Visualisasi

Kami memvisualisasikan beberapa variabel yang ada didalam dataset guna mencari besar pengaruh tiap variabel yang ada terhadap variabel target yaitu jumlah pemakaian sepeda tiap jam



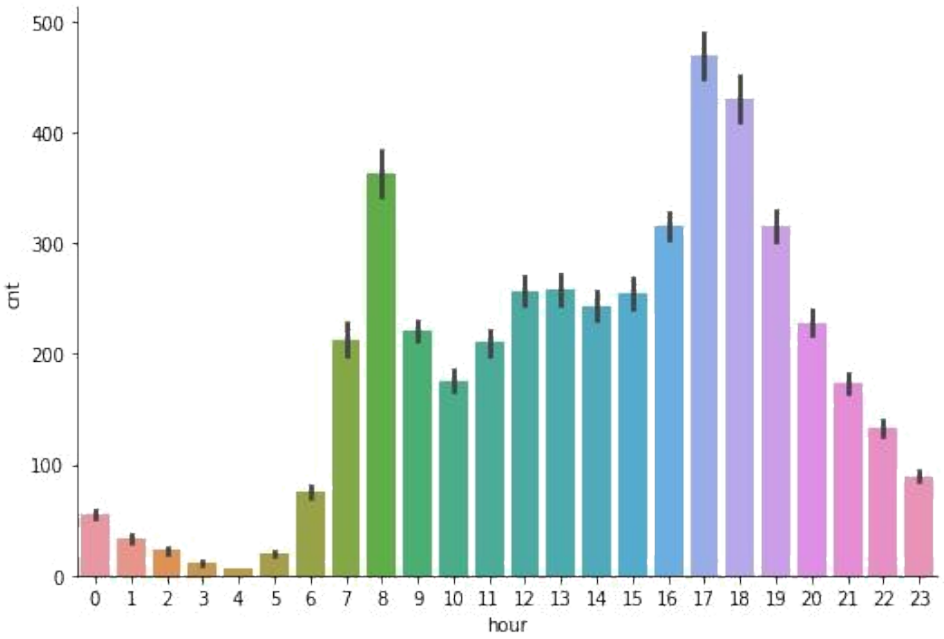
Gambar 1.1

Persebaran variabel temperature, kecepatan angin, dan kelembapan terhadap jumlah peminjaman sepeda



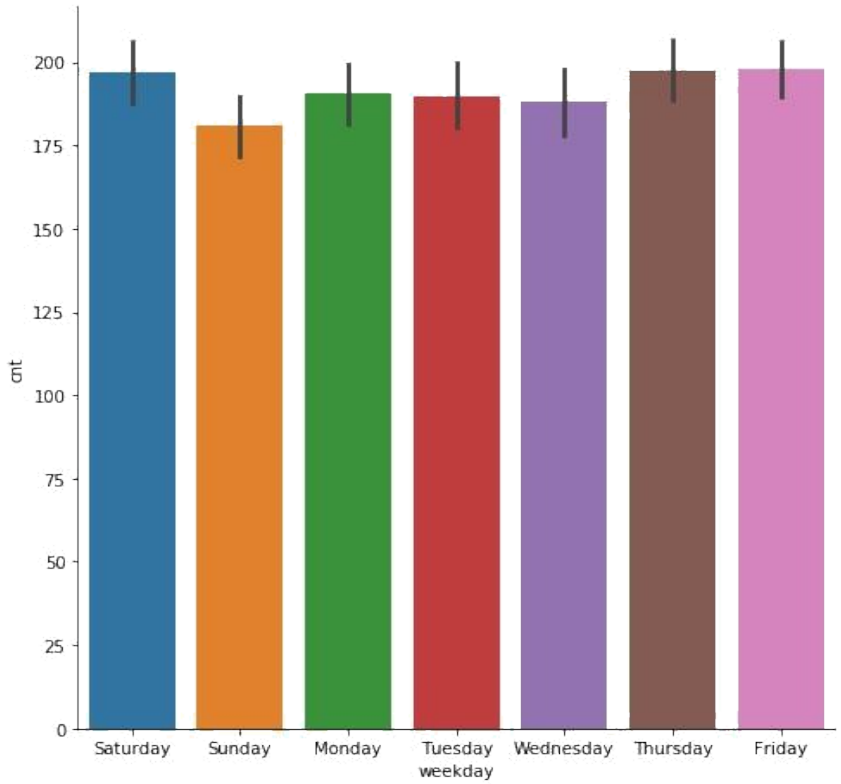
Gambar 1.2

Heatmap korelasi antar variabel



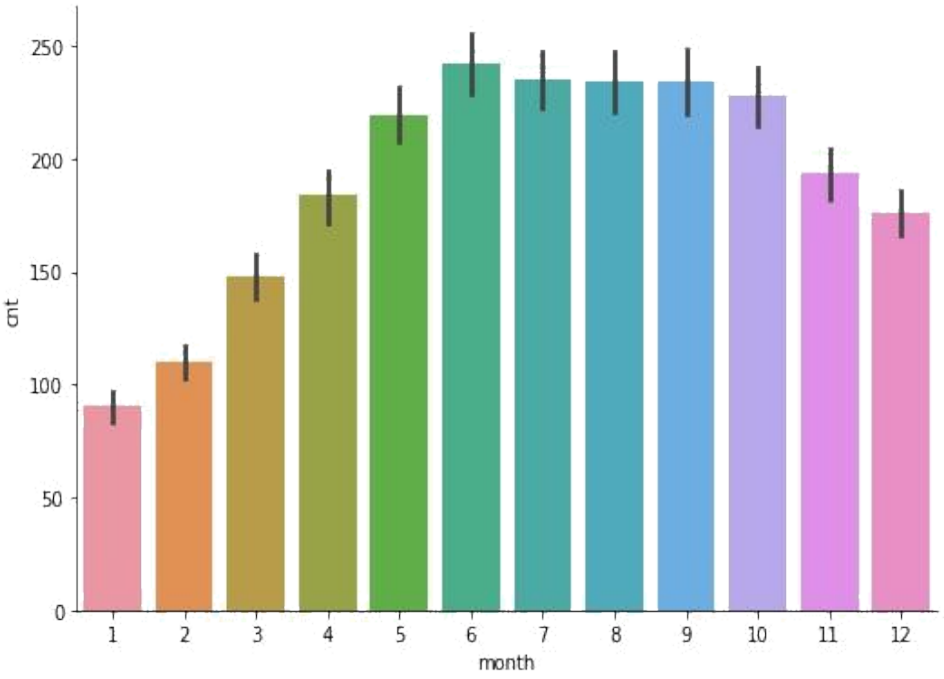
Gambar 1.3

Factorplot antara jam dan jumlah peminjaman sepeda



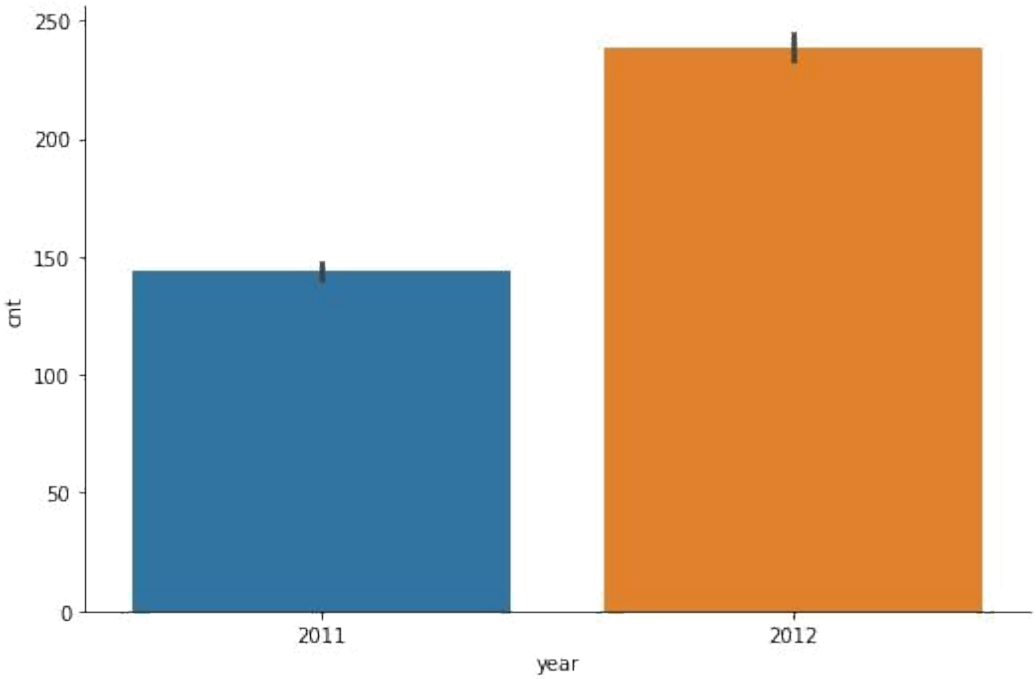
Gambar 1.4

Factorplot antar hari dengan jumlah peminjaman sepeda



Gambar 1.5

Factorplot antar bulan dengan jumlah peminjaman sepeda



Gambar 1.6

Factorplot antar tahun dengan jumlah peminjaman sepeda

C. Scalling Value

Kami menggunakan MinMaxScaler yang disediakan oleh scikit-learn untuk merubah (menormalisasikan) value sebelum masuk ke tahap pemodelan. Kami menormalisasi guna melandaikan simpangan yang terdapat pada variabel agar mendapatkan hasil yang terbaik padap pemodelan.

D. Modelling and Evaluation

Karena dataset dan masalah ini berjenis supervised kami menggunakan tiga teknik regressor untuk memperoleh prediksi jumlah peminjaman sepeda pada tiap jam.

1. Menggunakan Teknik Random Forest Regressor

   Metode Random Forest merupakan salah satu metode dalam Decision Tree. Random forest adalah kombinasi dari masing – masing tree yang baik kemudian dikombinasikan ke dalam satu model. Random Forest bergantung pada sebuah nilai vector random dengan distribusi yang sama pada semua pohon yang masing masing decision tree memiliki kedalaman yang maksimal. Random forest adalah classifier yang terdiri dari classifier yang berbentuk pohon {h(x, θ k), k = 1, . . .} dimana θk adalah random vector yang diditribusikan secara independen dan masing masing tree pada sebuah unit kan memilih class yang paling popular pada input x.

2. Menggunakan Teknik Bagging Regressor

   Bagging adalah singkatan dari bootstrap aggregating, menggunakan subdataset (bootstrap) untuk menghasilkan set pelatihan L (learning), L melatih dasar belajar menggunakan prosedur pembelajaran yang tidak stabil, dan kemudian, selama pengujian, mengambil rata-rata (Breiman, 1996). Bagging baik digunakan untuk klasifikasi dan regresi.

3. Menggunakan Teknik Gradient Boosting Regressor

   Gradient Boosting merupakan teknik dalam machine learning untuk masalah regresi dan klasifikasi yang menghasilkan model prediksi dalam bentuk gabungan (ensemble) model prediksi yang lemah. Pembangunan model dilakukan dengan menggunakan metode boosting, yaitu dengan membuat model baru untuk memprediksi error/residual dari model sebelumnya. Model baru ditambahkan hingga tidak ada lagi perbaikan pada error yang dapat dilakukan. Algoritme ini dinamakan gradient boosting karena menggunakan gradient descent untuk memperkecil error saat membuat model baru

HASIL

Untuk mendapatkan perkiraan jumlah penyewa, prediki dievaluasi menggunakan Root Mean SquaredLogarithmic Error (RMSLE), yang didefinisikan sebagai berikut

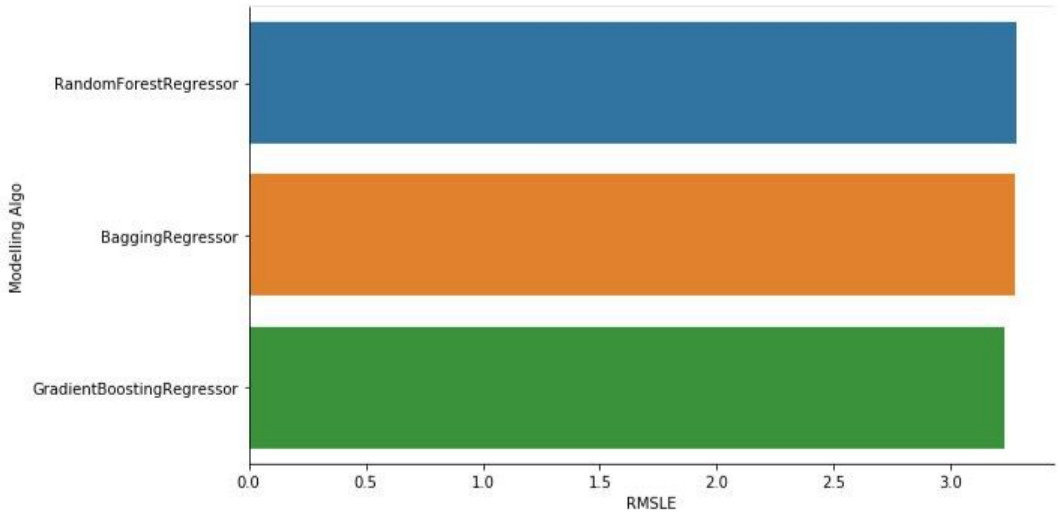
$$\mathbf{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\ln(p_i+1) - \ln(a_i+1))^2}$$

dimana n adalah Jumlah waktu dalam dataset test, p_i adalah nilai prediksi, a_i adalah nilai aktual dari , dan ln adalah natural logaritma.

Nilai RMSLE dengan model algoritma untuk perkiraan jumlah penyewa adalah sebagai berikut

	Modelling Algo	RMSLE
0	RandomForestRegressor	3.278070
1	BaggingRegressor	3.275049
2	GradientBoostingRegressor	3.230997

Tabel 1. Perbandingan Model Algoritma dengan RMSLE



Gambar 1. Perbandingan Model Algoritma dengan RMSLE

Dari grafik tersebut dapat simpulkan bahwa dengan menggunakan model Gradient Boosting Regressor memberikan hasil yang terkecil dengan nilai RMSLE sebesar 3.230997 didalam model dikarenakan terdapat kesalahan pada variabel uji . namun setelah di submit perhitungan prediksi yang terbaik adalah dari metode Random Forest Regressor yang mendapat 0.38066

KESIMPULAN

Setelah kami melakukan EDA (Exploratory Data Analysis) terhadap dataset ini kami dapat menyimpulkan bahwa para penggunaan sepeda lebih banyak pada hari kerja. Kami melihat bahwa puncak pemakaian sepeda terdapat pada pukul 17-18 dimana itu adalah jam pulang kantor(idealnya) dan pukul 8 dimana itu adalah jam berangkat kerja (idealnya) . dan dari semua variabel, variabel yang paling signifikan mempengaruhi jumlah penggunaan sepeda adalah waktu tiap jammnya. Dari 3 pemodelan algoritma diterapkan pada dataset bike share untuk memprediksi jumlah sepeda yang akan disewa per jam. Dalam kasus ini kami mendapat hasil dan galat dengan menggunakan algoritma Random Forest Regressor karena memiliki Root Mean Squared Logarithmic Error (RMSLE) paling kecil dibandingkan yang lainnya ketika di submit. Jadi, menurut laporan ini mengusulkan untuk memperkiraan jumlah sewa sepeda gunakan Random Forest Regressor sebagain permodelan algoritmanya.

PUSTAKA

Breiman L.2001.Random Forests.Machine Learning,45(1):5–32.

Breiman, L.1996.Bagging Predictors.Machine Learning, 123-140.

J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting," Annals of Statistics, vol. 38, no. 4, pp. 367-378, 2001.

Friedman, J. 2001. Greedy function approximation: a gradient boosting machine. An-nals of Statistics 29: 1189–1232.

LAMPIRAN

SOURCE CODE KERNEL PADA KAGGLE

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import os
print(os.listdir("../input"))

train = pd.read_csv('../input/bike-share-demand/train.csv')
test = pd.read_csv('../input/bike-share-demand/test.csv')
train_data = train.copy()
test_data = test.copy()

print(train_data.shape)
print(test_data.shape)

train_data.head()
train.describe()

train.isna().sum()

fig,(ax1,ax2,ax3,ax4) = plt.subplots(ncols=4)
fig.set_size_inches(12, 5)
sns.regplot(x="temp", y="cnt", data=train_data,ax=ax1)
sns.regplot(x="atemp", y="cnt", data=train_data,ax=ax2)
sns.regplot(x="windspeed", y="cnt", data=train_data,ax=ax3)
sns.regplot(x="humidity", y="cnt", data=train_data,ax=ax4)

cor= train[:].corr()
corval = np.array(cor)
corval[np.tril_indices_from(corval)] = False
fig=plt.gcf()
fig.set_size_inches(30,12)
sns.heatmap(data=cor,mask=corval,square=True,annot=True,cbar=True)

train_data['datetime'] = pd.to_datetime(train_data['datetime'])
train_data['weekday'] = train_data['datetime'].dt.weekday_name
train_data['year'] = train_data['datetime'].dt.year.astype(str)
train_data['hour'] = train_data['datetime'].dt.hour
train_data['month'] = train_data['datetime'].dt.month
test_data['datetime'] = pd.to_datetime(test_data['datetime'])
test_data['weekday'] = test_data['datetime'].dt.weekday_name
```

```
test_data['year'] =
test_data['datetime'].dt.year.astype(str)
test_data['hour'] = test_data['datetime'].dt.hour
test_data['month'] = test_data['datetime'].dt.month
train_data.columns

sns.factorplot(x="hour",y="cnt",data=train_data,kind='bar',size=5,aspect=
1.5)
def hour_group(s):
    if((0<=s) & (s<=6)):
        return 1
    elif((s==7) | (s==9)):
        return 2
    elif((s==8) | (s==16) | (s==19)):
        return 5
    elif((10<=s) & (s<=15)):
        return 4
    elif((s==17) | (s==18)):
        return 6
   elif(20<=s):
       return 3
train_data['hour_group'] =
train_data['hour'].apply(hour_group).astype(str)
test_data['hour_group'] = test_data['hour'].apply(hour_group).astype(str)

sns.factorplot(x="weekday",y='cnt',kind='bar',data=train_data,size=7,aspe
ct=1)

sns.factorplot(x="month",y="cnt",data=train_data,kind='bar',size=5,aspect
=1.5)

sns.factorplot(x="year",y="cnt",data=train_data,kind='bar',size=5,aspect=
1.5)

train_data['season'] = train_data['season'].astype(str)
test_data['season'] = test_data['season'].astype(str)
train_data['weather'] = train_data['weather'].astype(str)
test_data['weather'] = test_data['weather'].astype(str)
train_data['holiday'] = train_data['holiday'].astype(str)
test_data['holiday'] = test_data['holiday'].astype(str)
train_data['workingday'] = train_data['workingday'].astype(str)
test_data['workingday'] = test_data['workingday'].astype(str)

train_data = pd.get_dummies(train_data)
test_data = pd.get_dummies(test_data)

train_data.head()

x_train = train_data.copy()
del x_train['casual'], x_train['registered'], x_train['cnt'],
x_train['datetime'] , x_train['windspeed']
x_test = test_data.copy()
del x_test['datetime'], x_test['windspeed']
```

```
y_train = train_data['cnt']
y_test = test_data.copy()
y_test = pd.concat([test_data,train_data['cnt']], axis=1)
y_test = y_test.dropna(axis=0)
y_test = y_test['cnt']

from sklearn.preprocessing import
MinMaxScaler scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
y_train = np.log1p(y_train)
y_test = np.log1p(y_test)

from sklearn.ensemble import RandomForestRegressor
model =
                          RandomForestRegressor(n_estimator
                          s=1000,min_samples_leaf=2,
                          random_state=0, n_jobs=-1)
model.fit(x_train, y_train)
pred = model.predict(x_test)
pred = np.expm1(pred)
pred = np.round(pred)

from sklearn.metrics import
mean_squared_log_error rmsle =
np.sqrt(mean_squared_log_error(pred,y_test))
rmsle

from sklearn.ensemble import BaggingRegressor
model2 = BaggingRegressor(n_estimators=1000, random_state =0)
model2.fit(x_train,y_train)
pred2 = model2.predict(x_test)
pred2 = np.expm1(pred2)
pred2 = np.round(pred2)
rmsle2 = np.sqrt(mean_squared_log_error(pred2,y_test))
rmsle2/10

from sklearn import ensemble
model3 =
                                        ensemble.GradientBoostingRegresso
                                        r(max_features=10,learning_rate=0
                                        .01,
                                        n_estimators=1000,subsample=0.7,r
                                        ando
m_state=0)
model3.fit(x_train,y_train)
pred3 = model3.predict(x_test)
pred3 = np.expm1(pred3)
pred3 = np.round(pred3)
rmsle3 = np.sqrt(mean_squared_log_error(pred3,y_test))
rmsle3/10

output1 = pd.DataFrame({'datetime': test_data.datetime,
                        'count': pred})
```

```
output1.to_csv('submission.csv', index=False)

output2 = pd.DataFrame({'datetime': test_data.datetime,
                        'count': pred2})
output2.to_csv('submission2.csv', index=False)

output3 = pd.DataFrame({'datetime': test_data.datetime,
                        'count': pred3})
output3.to_csv('submission3.csv', index=False)
```