Makalah DSC ITToday 2019

QUADCOPTER FLIGHT PATH IDENTIFICATION

Sour Soup Team

Telkom University

Daftar Isi

1. Latar Belakang

Hazim sedang mengerjakan project kuliah yaitu merakit dua buah *quadcopter*. *Quadcopter* adalah helikopter multirotor yang diangkat dan didorong oleh empat rotor (baling-baling). Quadcopter dapat melakukan *take off* dan *landing* secara vertikal. *Vertical Take Off Landing* (VTOL) *Aircraft* merupakan jenis pesawat yang dapat melakukan *take off* dan landing tegak lurus terhadap bumi sehingga dapat dilakukan pada tempat yang sempit. *Quadcopter* yang dirakit Hazim diatur untuk terbang dengan dua tipe rute lintasan secara otomatis (*autopilot*) serta dapat dipantau melalui *remote control* ataupun komputer. Pada *quadcopter* tersebut terdapat beberapa sensor, diantaranya adalah *global positioning system* (GPS), *global position accuracy* (GPA), *inertial measurement unit* (IMU), dan magnetometer (Kompas).

GPS adalah sistem untuk menentukan letak di permukaan bumi dengan bantuan penyelarasan (*synchronization*) sinyal satelit. Selain itu, adalah sebuah perangkat elektronik yang mengukur dan melaporkan kecepatan kerajinan itu, orientasi, dan gaya gravitasi, menggunakan kombinasi akselerometer dan giroskop, kadang-kadang juga magnetometer. IMU biasanya digunakan untuk manuver pesawat, termasuk kendaraan udara tak berawak (UAV), antara lain banyak, dan pesawat ruang angkasa, termasuk satelit dan pendarat. Perkembangan terkini memungkinkan untuk produksi perangkat GPS IMU-*enabled*. Sebuah IMU memungkinkan penerima GPS untuk bekerja ketika GPS-sinyal tidak tersedia, seperti di terowongan, di dalam bangunan, atau ketika interferensi elektronik hadir. Sebuah IMU nirkabel dikenal sebagai WiMu.

Setelah tahap perakitan selesai, Hazim akan melakukan uji coba penerbangan pada *quadcopter* rakitannya untuk diambil datanya. Melalui kedua *quadcopter* tersebut, dilakukan uji coba terbang dengan mengikuti koordinat *longitude* dan *latitude* pada GPS *quadcopter* sesuai dengan rute yang telah diatur sebelumnya.

2. Tujuan dan Manfaat

Tujuan dari proses data mining ini adalah untuk membantu Hazim memprediksi data koordinat (*longitude*, *latitude*) deret waktu dari posisi ke-X hingga posisi terakhir untuk setiap percobaan berdasarkan 93 atribut atau *feature* yang telah ada pada *dataset* dan terbagi kedalam beberapa kelompok besar, yaitu: ID, ATT (*Attitude Information*), MAG/COMPASS (Raw Compass, Offset, and Compassmot Compensation Values), CTUN (Control, Throttle, and Altitude Information), GPA (*Global Position Accuracy*), GPS (*Global Positioning System*), IMU (*Inertial Measurement Unit*), dan RCOUT (RC *Outputs*).

3. Batasan – batasan

Kami akan memprediksi untuk 12 experimen dan terbagi 2 pattern yaitu persegi dan lingkaran seperti pada gambar 1. Selain itu, *latitude* dan *longitude* hanya dalam lingkup wilayah IPB (-6.557, 106.734).

Gambar 1 Gambaran *Pattern* dari Setiap *Record* Pada Data *Train*.

4. Metode Analisis Data

Data untuk men*train* model, kami menggunakan adalah keseluruhan data yang nilainya tidak NaN pada kolom ‘Lat’ dan ‘Lng’. Sebaliknya data untuk mengevaluasi model (data *test*), kami menggunakan adalah keseluruhan data yang nilainya NaN pada kolom ‘Lat’ dan ‘Lng’, serta terdapat pada file ‘QuadData.csv’ yang telah diberikan.

Untuk menyelesaikan permasalahan ini, algoritma yang diterapkan adalah metode *Extreme Gradient Boosting* (XGBoost), metode ini digunakan berdasarkan hasil terbaik setelah menggunakan *K-Nearest Neighbors* dan *Light Gradient Boosting Machine* (LightGBM). XGBoost 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. Selain itu XGBoost merupakan versi GBM yang lebih efisien dan scalable. XGBoost mampu mengerjakan berbagai fungsi seperti regresi, klasifikasi, dan ranking.

Adapun *software* yang digunakan untuk membangun model tersebut adalah jupyter notebook, dengan menggunakan Bahasa pemrograman python. Selain itu kami menggunakan beberapa *library* yang terdapat pada python tertera pada gambar 2.

```
## Data Analysis
import numpy as np
import pandas as pd
import seaborn as sns
import pandas_profiling as pp
import matplotlib.pyplot as plt
%matplotlib inline

## Machine Learning
import xgboost
from xgboost import XGBRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import ElasticNetCV, LassoCV, RidgeCV
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import KFold, cross_val_score, train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

## Other
import math
import warnings
from sklearn.preprocessing import LabelEncoder
```

Gambar 2 *Library* yang Digunakan Saat Menganalisis Data

Hasil dari model yang telah dibuat akan dievaluasi menggunakan metrik Haversine *distance*. Haversine *distance* pada umumnya digunakan dalam dunia navigasi. Rumus tersebut mengukur jarak antar dua titik berdasarkan *latitude* dan *longitude* suatu titik. Rumus Harvesine *distance* diantara dua titik adalah sebagai berikut:

$$a = sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + cos\left(\phi_1\right) cos\left(\phi_2\right) sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)$$

$$d = 2 \cdot r \cdot atan\left(\sqrt{\frac{a}{1-a}}\right)$$

Keterangan:

- ϕ adalah *latitude*
- λ adalah *longitude*
- d adalah jarak diantara dua titik
- r adalah radius bumi (6.371 km)

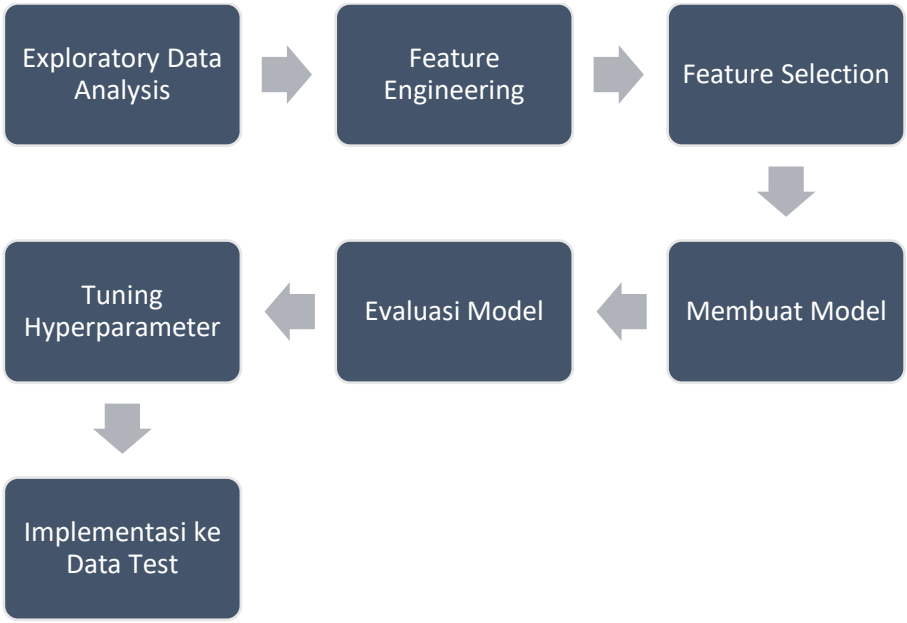
Adapun impelementasi kode dari haversine *distance* pada gambar 3.

```python
def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(math.radians, [lon1, lat1, lon2, lat2])

    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = math.sin(dlat/2)**2 + math.cos(lat1) * math.cos(lat2) * math.sin(dlon/2)**2
    c = 2 * math.asin(math.sqrt(a))
    r = 6371 # Radius of earth in kilometers. Use 3956 for miles
    return c
```

Gambar 3 Implementasi Haversine *Distance* Pada Kode

5. Desain dan Implementasi Analisis Data

Secara umum alur dan desain analisis yang kami buat terdapat pada gambar 4 di bawah ini.



Gambar 4 Alur Analisis Data

a. *Exploratory Data Analysis*

Pada tahap ini kami meneliti masing – masing *feature* atau atribut dari dataset, melihat korelasi antar *feature*, dan isi dari masing – masing *feature* dengan menggunakan *pandas-profiling* yang terdapat pada gambar 5.

```
diagnosis = pp.ProfileReport(df_train)
```

Gambar 5 Kode Untuk EDA dengan *Pandas-Profiling*

b. *Feature Engineering*

Kemudian kami melakukan explorasi dari data yang ada untuk dikembangkan atau merubah nilai dari setiap *feature* yang berhubungan dengan nilai dari *feature* tersebut.

c. *Feature Selection*

Setelah melakukan *feature engineering* kami melakukan seleksi *feature* untuk mengefisiensikan kinerja model sehingga model hanya membutuhkan *feature* yang penting untuk melakukan regresi dengan baik.

d. Membuat Model

Untuk awalan, kami membuat *base* model untuk melakukan regresi. Tahap ini merupakan pembuatan model awal dengan parameter *default* dari *tools* yang dipakai yakni XGBoost.

e. Evaluasi Model

Setelah melakukan pembuatan model, evaluasi hasil regresi yang telah dilakukan oleh model.

f. *Tuning Hyperparameter*

Mencari parameter model yang optimal agar performansi dari model dapat melakukan regresi dengan maksimal.

g. Implementasi Model ke Data *Test*

Setelah mencari parameter yang optimal, saatnya memprediksi koordinat (*latitude* dan *longitude*) pada data test.

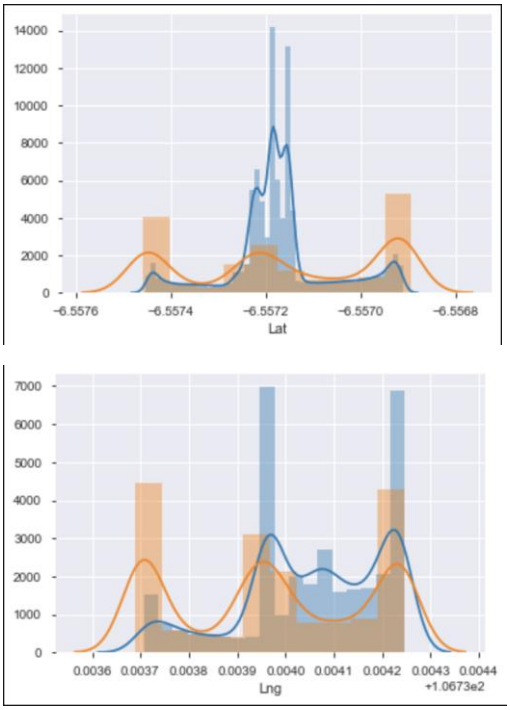
6. Analisis

a. *Feature* atau Atribut Pada Dataset

1. Kelas Target (y)

Pada kali ini kita akan melakukan regresi dua kali yakni terhadap *latitude* (‘Lat’) dan *longitude* (‘Lng’) sehingga ‘Lat’ dan ‘Lng’ adalah kelas target (y) untuk melakukan

prediksi koordinat *quadcopter*. Namun dikarenakan *range* dari ‘Lat’ dan ‘Lng’ kecil yang terlihat pada gambar 6.



Gambar 6 Distribusi Plot dari ‘Lat’ dan ‘Lng’

Oleh karena itu, kami putuskan untuk melakukan *scaling* dengan menggunakan metode *standard scaler* (z = (x - u) / s) pada kedua kelas target tersebut dengan kode pada gambar 7.

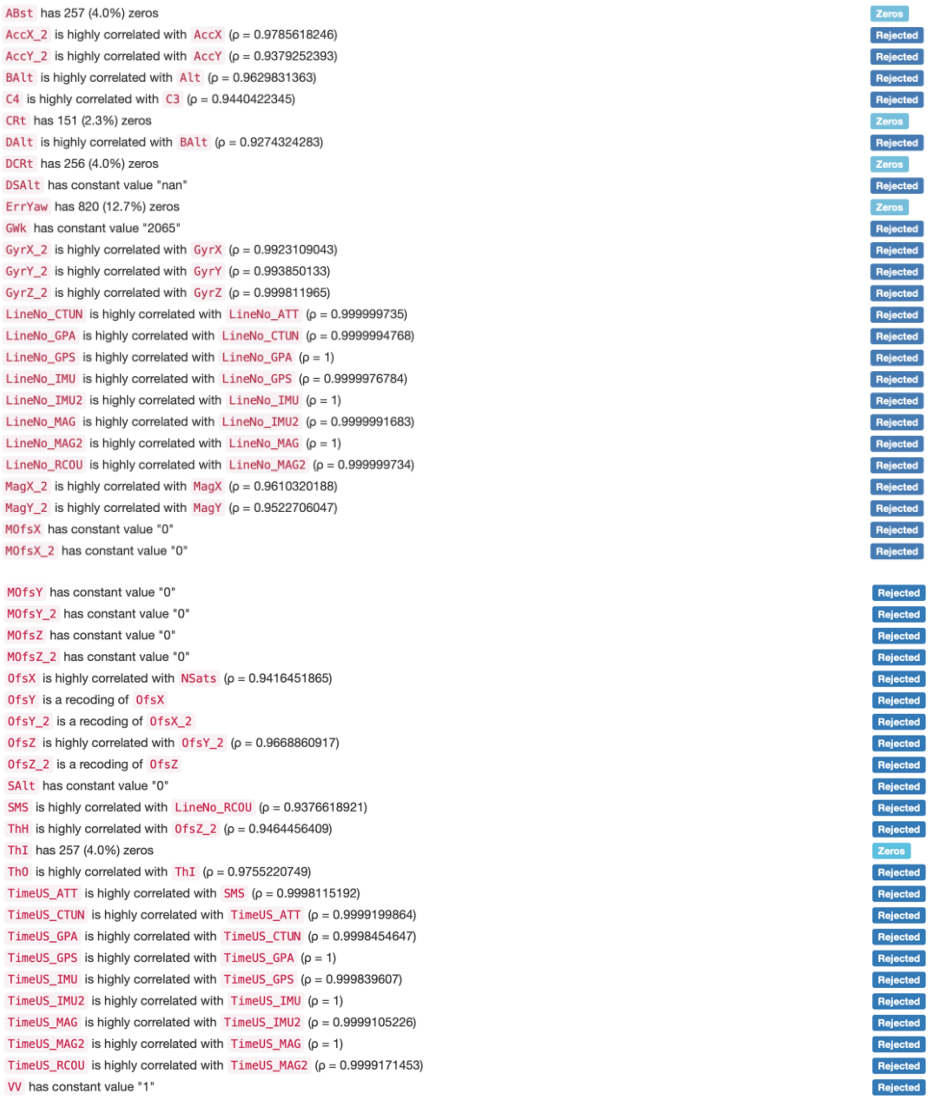
```
sc_lat = StandardScaler()
sc_lon = StandardScaler()

y_lat = sc_lat.fit_transform(df_train[['Lat']])
y_lon = sc_lon.fit_transform(df_train[['Lng']])
```

Gambar 7 *Scaling* 'Lat' dan 'Lng'

2. *Feature* atau atribut (X)

   Terdapat 93 *feature* atau atribut yang terbagi kedalam beberapa kelompok besar, yaitu: ID, ATT (*Attitude Information*), MAG/COMPASS (Raw Compass, Offset, and Compassmot Compensation Values), CTUN (Control, Throttle, and Altitude Information), GPA (*Global Position Accuracy*), GPS (*Global Positioning System*), IMU (*Inertial Measurement Unit*), dan RCOUT (RC *Outputs*). Adapun untuk menganalisis *feature* mana saja yang akan dipakai untuk membuat model, kami menggunakan *pandas-profiling* untuk melakukan *exploratory data analysis* (EDA) untuk melihat dengan pendekatan statistika terhadap masing – masing *feature*.

   Hasil dari *pandas-profiling* terdapat pada gambar 5. Untuk yang bertanda *'Rejected' pandas-profiling* menyarankan untuk tidak menjadikannya *feature* untuk model yang akan dibuat karena berkorelasi tinggi dengan *feature* tertentu, sedangakan untuk *'Zeros' pandas-profiling* memberikan peringatan karena *feature* tersebut memiliki nilai 0 yang tidak wajar. Hasil rekomendasi *pandas-profiling* secara umum digambarkan pada gambar 8.



Gambar 8 Hasil Rekomendasi EDA dari *Pandas-Profiling*

b. *Feature Engineering*

Pada *feature* 'PatternID' karena termasuk kedalam tipe object dan hanya memiliki 2 nilai unik, kami menjadikan *feature* tersebut menjadi variabel *dummy*, kodenya terdapat pada gambar 9.

```
df_train = pd.get_dummies(df_train)
```

Gambar 9 Membuat Variabel *Dummy* dari *Dataset*

Pada *feature engineering* kali ini kami membuat *feature* baru perbedaan antara *the pilot desired movement* (Roll, Pitch, Yaw) dan *actual*nya. Kemudian *feature* baru dari setiap *feature* degree ke nama arah mata angin dan menambahkan *feature* baru dari setiap

feature degree ke radian. Implementasi kode terdapat pada gambar 10.

Feature Engineering

```
In [132]: def degToCompass(num):
              val=int((num/22.5)+.5)
              arr=["N","NNE","NE","ENE","E","ESE", "SE", "SSE","S","SSW","SW","WSW","W","WNW","NW","NNW"]
              return arr[(val % 16)]

In [133]: def degree_to_radion(degree):
              return degree*(np.pi/180)

In [134]: for i in df_train.columns:
              if 'des' in i.lower():
                  df_train[i+'_Err'] = df_train[i] - df_train[i[3:]]

In [135]: for i in ['Pitch', 'Yaw', 'GCrs']:
              df_train[i+'_str'] = df_train[i].apply(degToCompass).astype('category')

In [136]: for i in ['Pitch', 'Yaw', 'GCrs']:
              df_train[i+'_radion'] = df_train[i].apply(degree_to_radion)
```

Gambar 10 *Feature Engineering* Pada *Dataset*

Penjelasan *feature engineering*

- line 134 : Asumsinya ada tempat atau belokan dimana menyebabkan perbedaan antara apa yang pilot inginkan dan actual nya.
- line 135 = degree secara raw sepertinya terlalu granular sehingga dikelompokkan menjadi nama mata angin dengan begitu model lebih generalize karena saat drone menghadap ke utara perbedaan beberapa derajat tidak terlalu berpengauh.
- line 136 : Karena radian lebih baik sebagai unit pengukur daripada degree

c. Membuat Model dan *Tuning Hyperparameter* Model

Pada sesi ini, kami melakukan pencarian *hyperparameter* yang optimal untuk model yang telah dibuat sebelumnya sehingga meningkatkan akurasi model sebelumnya. *Tools* yang digunakan untuk melakukan proses ini, kami menggunakan *Grid Search*. Parameter yang kami gunakan terdapat pada gambar 11.

```
parameters = {'nthread':[4], #when use hyperthread, xgboost may become slower
              'objective':['reg:linear'],
              'learning_rate': [.03, 0.05, .07], #so called `eta` value
              'max_depth': [5, 6, 7],
              'min_child_weight': [4],
              'silent': [1],
              'subsample': [0.7],
              'colsample_bytree': [0.7],
              'n_estimators': [500],
             }
```

Gambar 11 Parameter untuk Melakukan *Tuning Hyperparameter*

Kemudian melakukan proses pencarian parameter yang optimal untuk meningkatkan performansi dari model yang telah dibuat.

```
xgb_grid_lat = GridSearchCV(xgb1,
                            parameters,
                            cv = 10,
                            n_jobs = -1,
                            verbose=True)

xgb_grid_lon = GridSearchCV(xgb1,
                            parameters,
                            cv = 10,
                            n_jobs = -1,
                            verbose=True)

xgb_grid_lat.fit(X,y_lat)
xgb_grid_lon.fit(X,y_lon)
```

Gambar 12 Melakukan Pencarian *Hyperparameter* yang Optimal

7. Kesimpulan

Proses data mining yang kami lakukan dimulai dengan melakukan proses *exploratory data analysis* (analisis dataset) yang terdiri dari proses mengatasi *missing data, feature engineering, dan feature selection*. Setelah itu kami lakukan *modelling* dengan menggunakan metode *Extreme Gradient Boosting* (XGBoost). Hasil pemodelan diterapkan pada data *test* untuk dilakukan regresi terhadap kedua kelas target y (*latitude* dan *longitude*) pada data *test* tersebut. Pada akhirnya, hasil prediksi regresi dengan model yang kami bangun disimpan pada file ‘submmission.csv’.

8. Dokumentasi Kegiatan