Machine Learning Project

Predicting A Pulsar Star

Submitted By :

Steven - 13433
B. Bias A. Ch. - 13536
Raynaldi - 13654

Contents

1 Identifikasi Masalah **2**

2 Analisis Dataset **3**
2.1 Informasi Dataset 3
2.2 Exploratory Data Analysis 4
2.3 Data Preprocessing 10

3 Evaluasi Model **11**
3.1 Logistic Regression 11
3.2 Decision Tree Classifier 12
3.3 Random Forest Classifier 13
3.4 Naive Bayes Classifier 14
3.5 K Nearest Neighbor 15
3.6 Support Vector Machine 16

4 Kesimpulan **17**
4.1 Confusion Matrix 17
4.2 Perbandingan Akurasi 18
4.3 Tingkat Kesalahan Klasifikasi 18

5 Kontribusi Anggota **19**

1 Identifikasi Masalah

HTRU2 adalah kumpulan data yang menggambarkan sampel kandidat pulsar yang dikumpulkan selama Survei Alam Semesta Resolusi Tinggi. [1]

Pulsar adalah jenis bintang Neutron yang langka yang menghasilkan emisi radio yang dapat dideteksi di Bumi. Mereka sangat menarik secara ilmiah sebagai wahana ruang-waktu, medium antar-bintang, dan keadaan materi.

Setiap pulsar menghasilkan pola emisi yang sedikit berbeda, yang sedikit berbeda pada setiap putaran. Dengan demikian deteksi sinyal potensial yang dikenal sebagai 'kandidat', dirata-ratakan pada banyak rotasi pulsar, sebagaimana ditentukan oleh panjang pengamatan. Dengan tidak adanya info tambahan, masing-masing kandidat berpotensi menggambarkan pulsar yang sebenarnya. Namun dalam praktiknya hampir semua deteksi disebabkan oleh interferensi frekuensi radio (RFI) dan kebisingan, membuat sinyal yang sah sulit ditemukan.

Alat pembelajaran mesin sekarang digunakan untuk secara otomatis melabeli kandidat pulsar untuk memfasilitasi analisis yang cepat. Sistem klasifikasi khususnya sedang banyak diadopsi, yang memperlakukan set data kandidat sebagai masalah klasifikasi biner. Di sini contoh pulsar yang sah adalah kelas positif minoritas, dan contoh palsu adalah kelas negatif mayoritas.

Kumpulan data yang dibagikan di sini berisi 16.259 contoh palsu yang disebabkan oleh RFI / noise, dan 1.639 contoh pulsar nyata. Semua contoh ini telah diperiksa oleh annotator manusia.

Setiap baris mencantumkan variabel terlebih dahulu, dan label kelas adalah entri terakhir. Label kelas yang digunakan adalah 0 (negatif) dan 1 (positif).

2 Analisis Dataset

2.1 Informasi Dataset

Setiap kandidat dijelaskan oleh 8 variabel kontinu, dan variabel kelas tunggal. Empat yang pertama adalah statistik sederhana yang diperoleh dari profil pulsa terintegrasi (profil terlipat). Ini adalah array variabel kontinu yang menggambarkan versi sinyal bujur yang diselesaikan yang telah dirata-rata dalam waktu dan frekuensi. Empat variabel sisanya diperoleh dengan cara yang sama dari kurva DM-SNR ini dirangkum di bawah ini:

1. Mean of the integrated profile
   Gelombang dari sebuah pulsar periodik emisi, rata-rata serentak selama beberapa ratus pulsa atau lebih. Juga dikenal sebagai profil terintegrasi.
2. Standard deviation of the integrated profile.
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
   Ukuran dari asimetri distribusi. Bisa nol jika distribusinya simetris, tetapi juga negatif atau positif.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.
7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR curve.
9. Class.

Ringkasan HTRU2 17.898 total contoh.
1.639 contoh positif. 16.259 contoh negatif.

2.2 Exploratory Data Analysis

1. Loading Dataset

```
DataFrame = pd.read_csv("datasets/pulsar_stars.csv")
DataFrame.head()
```

	Mean of the integrated profile	Standard deviation of the integrated profile	Excess kurtosis of the integrated profile	Skewness of the integrated profile	Mean of the DM-SNR curve	Standard deviation of the DM-SNR curve	Excess kurtosis of the DM-SNR curve	Skewness of the DM-SNR curve	target_class
0	140.562500	55.683782	-0.234571	-0.699648	3.199833	19.110426	7.975532	74.242225	0
1	102.507812	58.882430	0.465318	-0.515088	1.677258	14.860146	10.576487	127.393580	0
2	103.015625	39.341649	0.323328	1.051164	3.121237	21.744669	7.735822	63.171909	0
3	136.750000	57.178449	-0.068415	-0.636238	3.642977	20.959280	6.896499	53.593661	0
4	88.726562	40.672225	0.600866	1.123492	1.178930	11.468720	14.269573	252.567306	0

2. Cek apakah ada data yang null

```
# information about data types and amount of non-null rows of our Dataset
DataFrame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17898 entries, 0 to 17897
Data columns (total 9 columns):
 Mean of the integrated profile                17898 non-null float64
 Standard deviation of the integrated profile  17898 non-null float64
 Excess kurtosis of the integrated profile     17898 non-null float64
 Skewness of the integrated profile            17898 non-null float64
 Mean of the DM-SNR curve                      17898 non-null float64
 Standard deviation of the DM-SNR curve        17898 non-null float64
 Excess kurtosis of the DM-SNR curve           17898 non-null float64
 Skewness of the DM-SNR curve                  17898 non-null float64
target_class                                   17898 non-null int64
dtypes: float64(8), int64(1)
memory usage: 1.2 MB
```

```
DataFrame.isnull().sum()
```

```
 Mean of the integrated profile                0
 Standard deviation of the integrated profile  0
 Excess kurtosis of the integrated profile     0
 Skewness of the integrated profile            0
 Mean of the DM-SNR curve                      0
 Standard deviation of the DM-SNR curve        0
 Excess kurtosis of the DM-SNR curve           0
 Skewness of the DM-SNR curve                  0
target_class                                   0
dtype: int64
```

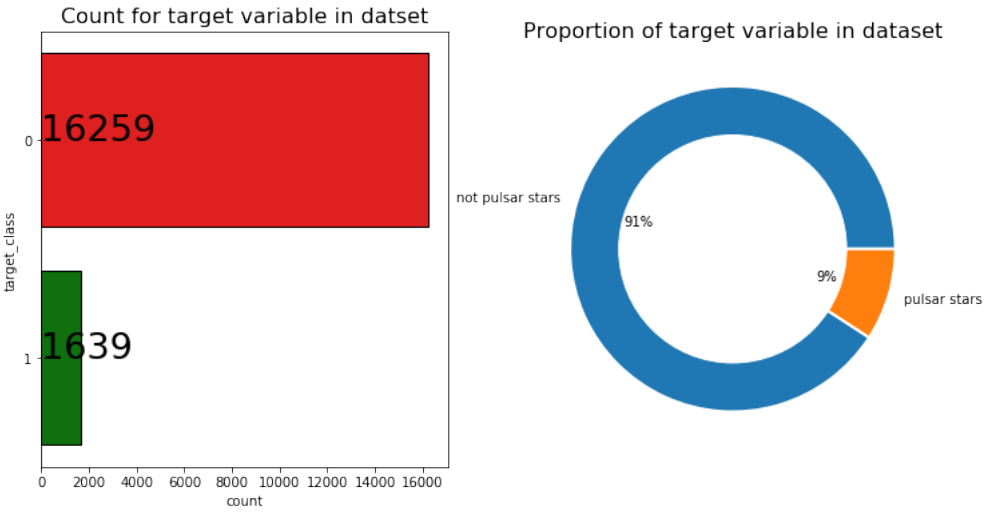
Karena tidak ada data yang null, tidak diperlukan transformasi data ataupun pembersihan data.

3. Lihat data statistik

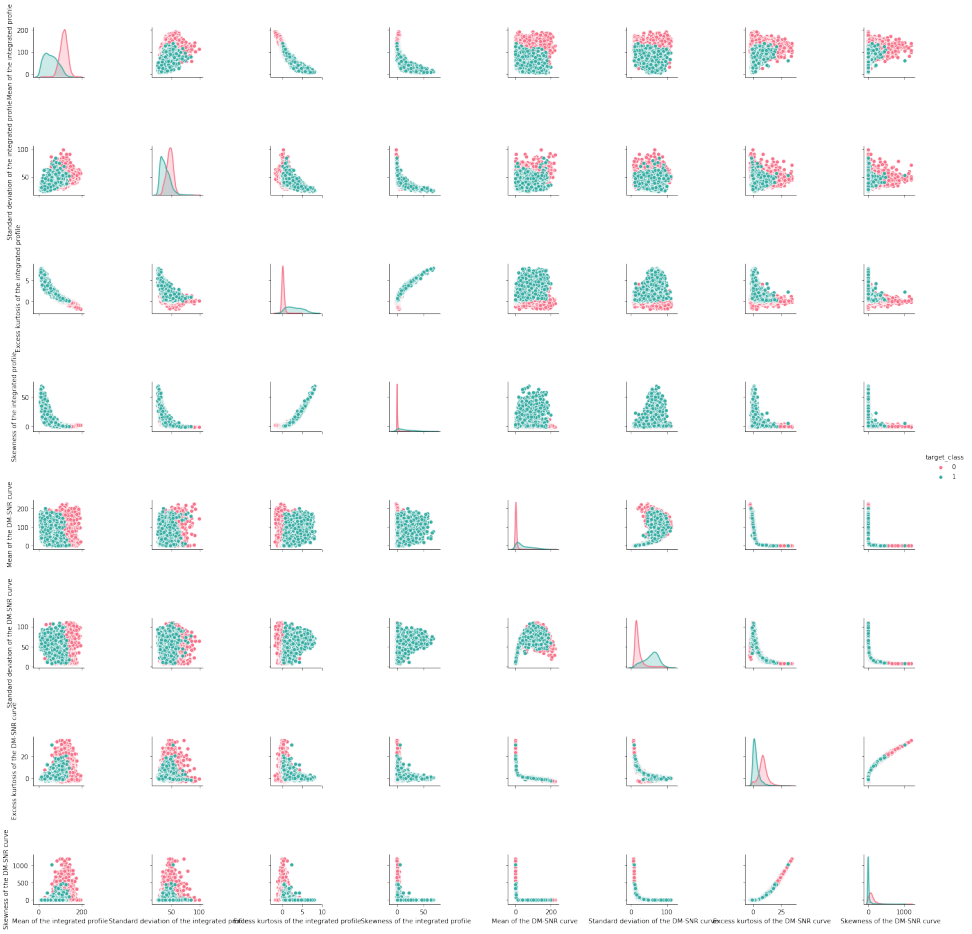
```
# statistical information about our data
DataFrame.describe()
```

	Mean of the integrated profile	Standard deviation of the integrated profile	Excess kurtosis of the integrated profile	Skewness of the integrated profile	Mean of the DM-SNR curve	Standard deviation of the DM-SNR curve	Excess kurtosis of the DM-SNR curve	Skewness of the DM-SNR curve	target_class
count	17898.000000	17898.000000	17898.000000	17898.000000	17898.000000	17898.000000	17898.000000	17898.000000	17898.000000
mean	111.079968	46.549532	0.477857	1.770279	12.614400	26.326515	8.303556	104.857709	0.091574
std	25.652935	6.843189	1.064040	6.167913	29.472897	19.470572	4.506092	106.514540	0.288432
min	5.812500	24.772042	-1.876011	-1.791886	0.213211	7.370432	-3.139270	-1.976976	0.000000
25%	100.929688	42.376018	0.027098	-0.188572	1.923077	14.437332	5.781506	34.960504	0.000000
50%	115.078125	46.947479	0.223240	0.198710	2.801839	18.461316	8.433515	83.064556	0.000000
75%	127.085938	51.023202	0.473325	0.927783	5.464256	28.428104	10.702959	139.309331	0.000000
max	192.617188	98.778911	8.069522	68.101622	223.392140	110.642211	34.539844	1191.000837	1.000000

4. Perbandingan porsi Target



5. Korelasi variabel data



Kita dapat melihat bahwa data kita cukup dapat dipisahkan pada sebagian besar kolom.

Sebagian besar kolom sudah terkait atau berasal dari satu atau yang lain, dan kita bisa melihatnya dengan jelas pada beberapa sel di atas.

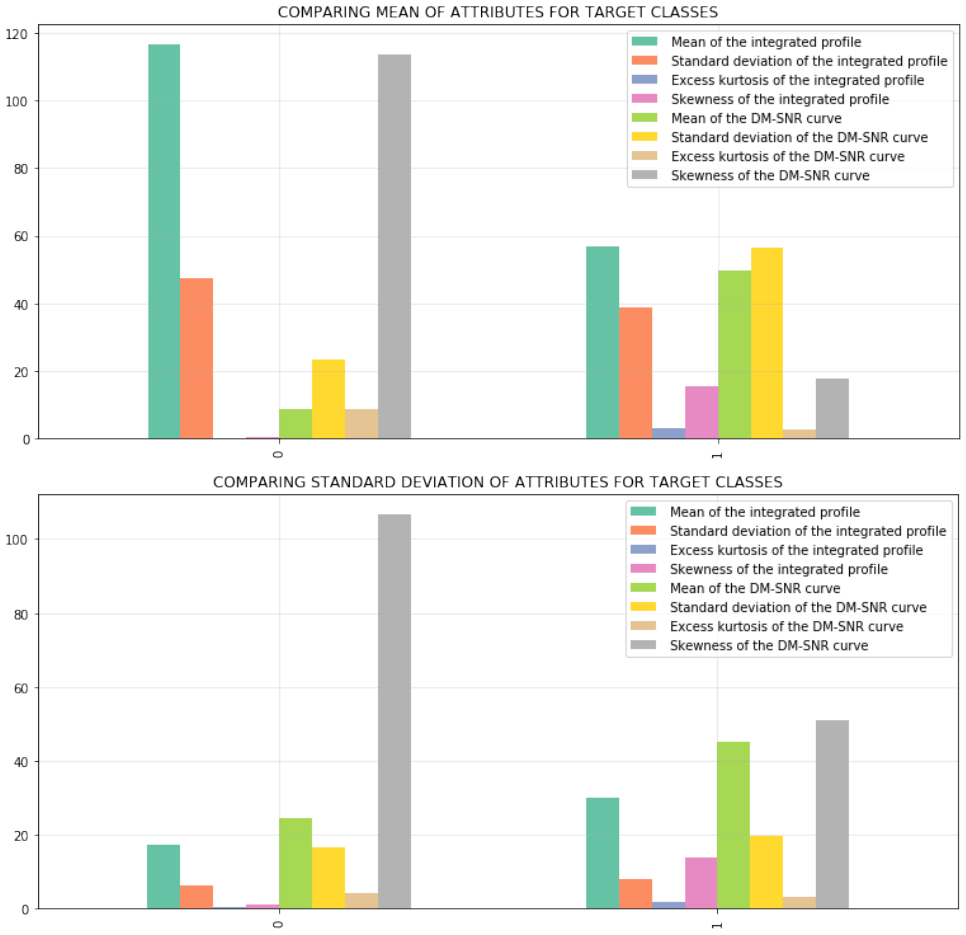
6. Membandingkan Rata-Rata dengan Standar Deviasi

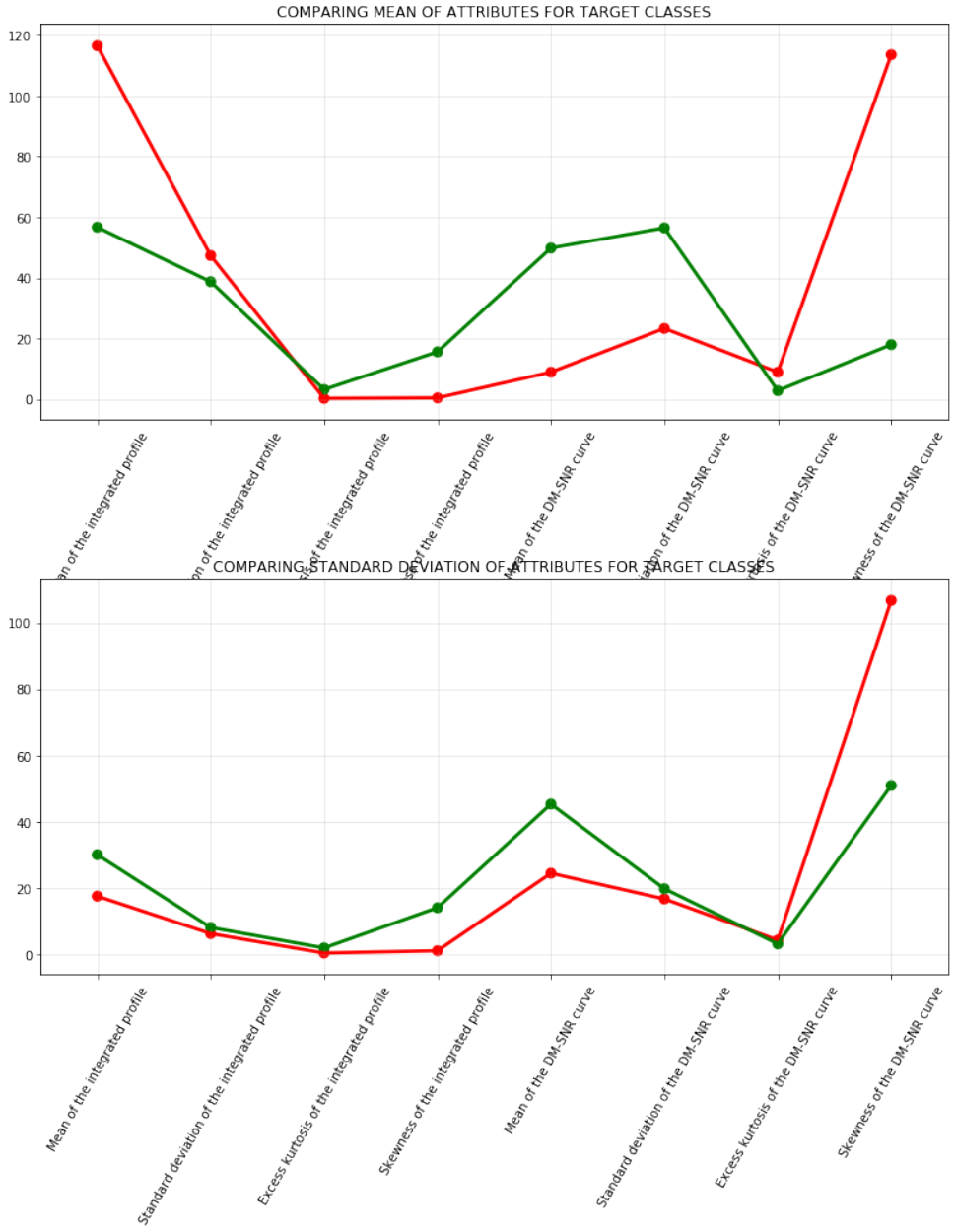
   (a) Rata-Rata

	features	not_star	star
0	Mean of the integrated profile	116.562726	56.690608
1	Standard deviation of the integrated profile	47.339741	38.710598
2	Excess kurtosis of the integrated profile	0.210440	3.130655
3	Skewness of the integrated profile	0.380844	15.553576
4	Mean of the DM-SNR curve	8.863258	49.825995
5	Standard deviation of the DM-SNR curve	23.287984	56.468963
6	Excess kurtosis of the DM-SNR curve	8.862674	2.757069
7	Skewness of the DM-SNR curve	113.620344	17.931728

(b) Standar Deviasi

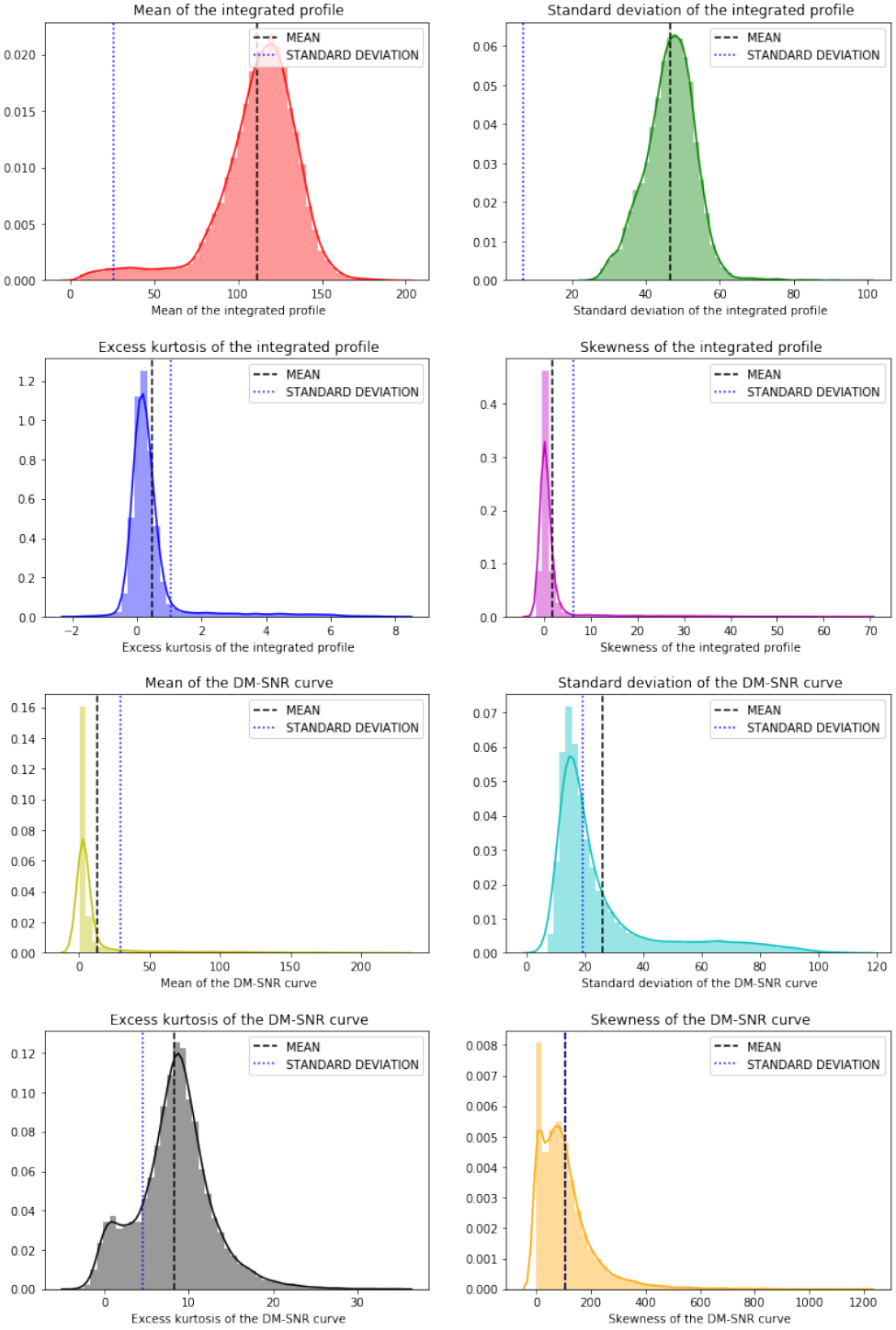
	features	not_star	star
0	Mean of the integrated profile	17.475932	30.007707
1	Standard deviation of the integrated profile	6.182929	8.033614
2	Excess kurtosis of the integrated profile	0.334606	1.872861
3	Skewness of the integrated profile	1.027791	13.997200
4	Mean of the DM-SNR curve	24.411409	45.287932
5	Standard deviation of the DM-SNR curve	16.651426	19.731080
6	Excess kurtosis of the DM-SNR curve	4.238626	3.105945
7	Skewness of the DM-SNR curve	106.721930	50.896263





Hijau = Bintang, Merah = Bukan Bintang

7. Melihat Distribusi dari variabel



2.3 Data Preprocessing

1. Memisahkan Fitur dengan Target

```
targets = DataFrame.target_class.values
print("Labels: %s\n" % targets)

features = DataFrame.drop(
    ["target_class"],
    axis=1,
)
print("Features: \n%s" % features)
```

```
Labels: [0 0 0 ... 0 0 0]

Features:
       Mean of the integrated profile  \
0                           140.562500
1                           102.507812
2                           103.015625
3                           136.750000
4                            88.726562
5                            93.570312
6                           119.484375
7                           130.382812
8                           107.250000
9                           107.257812
10                          142.078125
11                          133.257812
12                          134.960938
13                          117.945312
14                          138.179688
```

2. Mengskalasikan Fitur yang ada dengan MinMaxScalar

```
from sklearn.preprocessing import MinMaxScaler

minMaxScaler = MinMaxScaler(feature_range=(0, 1))
features_scaled_minmax = minMaxScaler.fit_transform(features)
print("Features Scaled By MinMax: \n%s \n" % features_scaled_minmax)
```

```
Features Scaled By MinMax:
[[0.72134164 0.41768745 0.16504291 ... 0.11368057 0.29498574 0.06388987]
 [0.51762787 0.46090841 0.23541516 ... 0.0725243  0.36401483 0.10844339]
 [0.52034628 0.19686832 0.22113842 ... 0.13918843 0.28862387 0.05461031]
 ...
 [0.60771193 0.4751437  0.2046521  ... 0.49869934 0.14965285 0.00550903]
 [0.58186609 0.39361695 0.20885482 ... 0.05820853 0.34892638 0.11418141]
 [0.27435072 0.82458965 0.33003783 ... 0.5552546  0.04091771 0.00285542]]
```

3 Evaluasi Model

Sebelum menggunakan model, kita akan mencari parameter yang dapat dianggap terbaik menggunakan GridSearch dari SciKit-Learn.

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import GridSearchCV
```

3.1 Logistic Regression

Parameter yang akan dicari adalah: ‘C‘

```
from sklearn.linear_model import LogisticRegression

# Setup the hyperparameter grid
logreg_param = {
    'random_state': [42],
    'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
    'solver': ["liblinear"],
    'penalty': ["l1"]
}

logreg = LogisticRegression()
logreg_cv = GridSearchCV(logreg, logreg_param, cv=5)
logreg_cv.fit(x_train, y_train)

print("Best Logistic Regression Estimator: {}\n".format(logreg_cv.best_estimator_))
print("Best Logistic Regression Score: {}".format(logreg_cv.best_score_))
```

```
Best Logistic Regression Estimator: LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l1',
                   random_state=42, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)

Best Logistic Regression Score: 0.9786932876406168
```

Setelah didapatkan best parameternya, gunakan ke dalam model.

```
# Use model from best estimator
lr_model = logreg_cv.best_estimator_

lr_model.fit(x_train, y_train)
y_head_lr = lr_model.predict(x_test)

lr_accuracy = accuracy_score(y_test, y_head_lr)
lr_report = classification_report(y_test, y_head_lr)
cm_lr = confusion_matrix(y_test, y_head_lr)

print("Logistic Regression Accuracy: {}\n".format(lr_accuracy))
print("Logistic Regression Report: \n{}".format(lr_report))
```

```
Logistic Regression Accuracy: 0.9805586592178771

Logistic Regression Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      4057
           1       0.94      0.84      0.89       418

    accuracy                           0.98      4475
   macro avg       0.96      0.92      0.94      4475
weighted avg       0.98      0.98      0.98      4475
```

3.2 Decision Tree Classifier

Parameter yang akan dicari adalah: ‘max_depth‘

```
from sklearn.tree import DecisionTreeClassifier

# Setup the hyperparameter grid
dectree_param = {
    'random_state': [42],
    'max_depth': np.arange(1, 100)
}

dectree = DecisionTreeClassifier()
dectree_cv = GridSearchCV(dectree, dectree_param, cv=5)
dectree_cv.fit(x_train, y_train)

print("Best Decision Tree Classifier Estimator: {}\n".format(dectree_cv.best_estimator_))
print("Best Decision Tree Classifier Score: {}".format(dectree_cv.best_score_))
```

```
Best Decision Tree Classifier Estimator: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=4,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=42, splitter='best')

Best Decision Tree Classifier Score: 0.9773523057438724
```

Setelah didapatkan best parameternya, gunakan ke dalam model.

```
# Use model from best estimator
dc_model = dectree_cv.best_estimator_

dc_model.fit(x_train, y_train)
y_head_dc = dc_model.predict(x_test)

dc_accuracy = accuracy_score(y_test, y_head_dc)
dc_report = classification_report(y_test, y_head_dc)
cm_dc = confusion_matrix(y_test, y_head_dc)

print("Decision Tree Classifier Accuracy: {}\n".format(dc_accuracy))
print("Decision Tree Classifier Report: \n{}".format(dc_report))
```

```
Decision Tree Classifier Accuracy: 0.9796648044692737

Decision Tree Classifier Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      4057
           1       0.92      0.85      0.89       418

    accuracy                           0.98      4475
   macro avg       0.95      0.92      0.94      4475
weighted avg       0.98      0.98      0.98      4475
```

3.3 Random Forest Classifier

Parameter yang akan dicari adalah: ‘n_estimators‘

```
from sklearn.ensemble import RandomForestClassifier

# Setup the hyperparameter grid
ranfor_param = {
    'n_estimators': np.arange(1, 100),
    'random_state': [42],
    'max_leaf_nodes': [200],
    'criterion': ["gini", "entropy"]
}

ranfor = RandomForestClassifier()
ranfor_cv = GridSearchCV(ranfor, ranfor_param, cv=5)
ranfor_cv.fit(x_train, y_train)

print("Best Random Forest Classifier Estimator: {}\n".format(ranfor_cv.best_estimator_))
print("Best Random Forest Classifier Score: {}".format(ranfor_cv.best_score_))
```

```
Best Random Forest Classifier Estimator: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=200,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=62,
                       n_jobs=None, oob_score=False, random_state=42, verbose=0,
                       warm_start=False)

Best Random Forest Classifier Score: 0.9804812635029427
```

Setelah didapatkan best parameternya, gunakan ke dalam model.

```
# Use model from best estimator
rfc_model = ranfor_cv.best_estimator_

rfc_model.fit(x_train, y_train)
y_head_rfc = rfc_model.predict(x_test)

rfc_accuracy = accuracy_score(y_test, y_head_rfc)
rfc_report = classification_report(y_test, y_head_rfc)
cm_rfc = confusion_matrix(y_test, y_head_rfc)

print("Random Forest Classifier Accuracy: {}\n".format(rfc_accuracy))
print("Random Forest Classifier Report: \n{}".format(rfc_report))
```

```
Random Forest Classifier Accuracy: 0.9796648044692737

Random Forest Classifier Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      4057
           1       0.92      0.85      0.89       418

    accuracy                           0.98      4475
   macro avg       0.95      0.92      0.94      4475
weighted avg       0.98      0.98      0.98      4475
```

3.4 Naive Bayes Classifier

Tidak ada parameter yang dapat di tune-up

```python
from sklearn.naive_bayes import GaussianNB
nb_model = GaussianNB()

nb_model.fit(x_train, y_train)
y_head_nb = nb_model.predict(x_test)

nb_accuracy = accuracy_score(y_test, y_head_nb)
nb_report = classification_report(y_test, y_head_nb)
cm_nb = confusion_matrix(y_test, y_head_nb)

print("Naive Bayes Classifier Estimator: {}\n".format(nb_model))
print("Naive Bayes Classifier Accuracy: {}\n".format(nb_accuracy))
print("Naive Bayes Classifier Report: \n{}".format(nb_report))
```

```
Naive Bayes Classifier Estimator: GaussianNB(priors=None, var_smoothing=1e-09)

Naive Bayes Classifier Accuracy: 0.9501675977653631

Naive Bayes Classifier Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      4057
           1       0.69      0.85      0.76       418

    accuracy                           0.95      4475
   macro avg       0.84      0.90      0.87      4475
weighted avg       0.96      0.95      0.95      4475
```

3.5 K Nearest Neighbor

Parameter yang akan dicari adalah: ‘n_neighbors’, ’weights‘

```
from sklearn.neighbors import KNeighborsClassifier

# Setup the hyperparameter grid
knn_param = {
    'n_neighbors': np.arange(1, 10),
    'weights': ["uniform", "distance"]
}

knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn, knn_param, cv=5)
knn_cv.fit(x_train, y_train)

print("Best K Nearest Neighbor Estimator: {}\n".format(knn_cv.best_estimator_))
print("Best K Nearest Neighbor Score: {}".format(knn_cv.best_score_))
```

```
Best K Nearest Neighbor Estimator: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                     weights='distance')

Best K Nearest Neighbor Score: 0.9789912836176712
```

Setelah didapatkan best parameternya, gunakan ke dalam model.

```
# Use model from best estimator
knn_model = knn_cv.best_estimator_

knn_model.fit(x_train, y_train)
y_head_knn = knn_model.predict(x_test)

knn_accuracy = accuracy_score(y_test, y_head_knn)
knn_report = classification_report(y_test, y_head_knn)
cm_knn = confusion_matrix(y_test, y_head_knn)

print("K Nearest Neighbor Accuracy: {}\n".format(knn_accuracy))
print("K Nearest Neighbor Report: \n{}".format(knn_report))
```

```
K Nearest Neighbor Accuracy: 0.980782122905028

K Nearest Neighbor Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      4057
           1       0.95      0.84      0.89       418

    accuracy                           0.98      4475
   macro avg       0.97      0.92      0.94      4475
weighted avg       0.98      0.98      0.98      4475
```

3.6 Support Vector Machine

Parameter yang akan dicari adalah: ‘C‘

```
from sklearn.svm import SVC

# Setup the hyperparameter grid
svm_param = {
    'random_state': [42],
    'C': np.arange(50, 300, 25),
    'gamma': [1.6],
    'kernel': ["poly"],
    'probability': [True]
}

svm = SVC()
svm_cv = GridSearchCV(svm, svm_param, cv=5)
svm_cv.fit(x_train, y_train)

print("Best Support Vector Machine Estimator: {}\n".format(svm_cv.best_estimator_))
print("Best Support Vector Machine Score: {}".format(svm_cv.best_score_))
```

```
Best Support Vector Machine Estimator: SVC(C=50, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=1.6, kernel='poly',
    max_iter=-1, probability=True, random_state=42, shrinking=True, tol=0.001,
    verbose=False)

Best Support Vector Machine Score: 0.9790657826119348
```

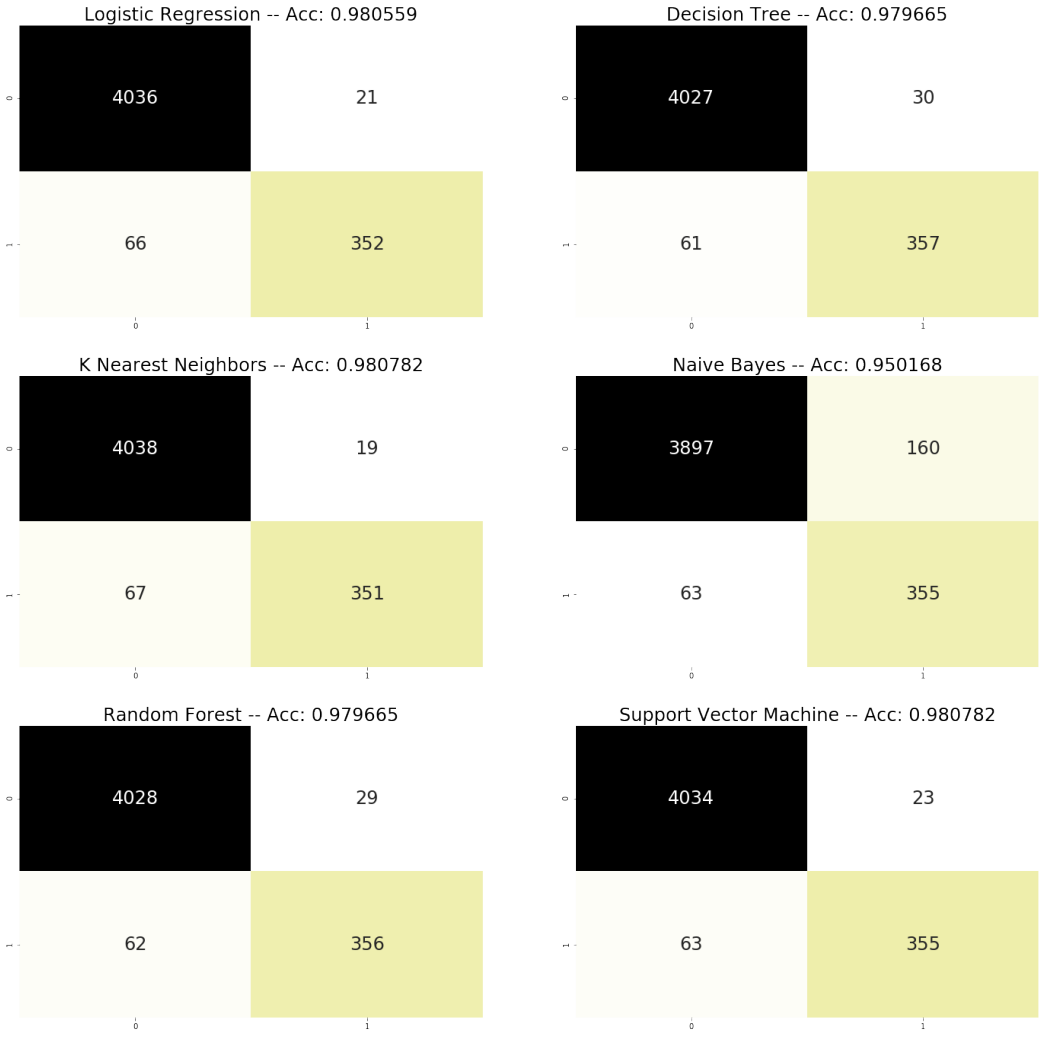
Setelah didapatkan best parameternya, gunakan ke dalam model.

```
# Use model from best estimator
svm_model = svm_cv.best_estimator_

svm_model.fit(x_train, y_train)
y_head_svm = svm_model.predict(x_test)

svm_accuracy = accuracy_score(y_test, y_head_svm)
svm_report = classification_report(y_test, y_head_svm)
cm_svm = confusion_matrix(y_test, y_head_svm)

print("Support Vector Machine Accuracy: {}\n".format(svm_accuracy))
print("Support Vector Machine Report: \n{}".format(svm_report))
```

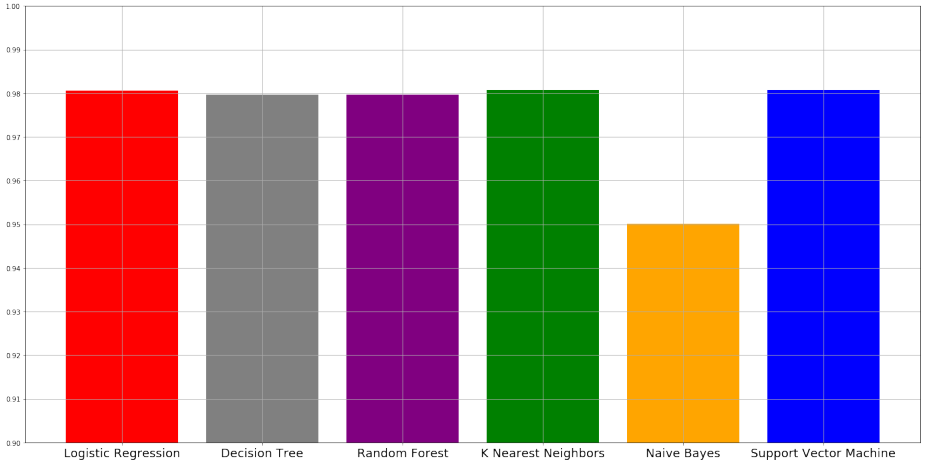
```
Support Vector Machine Accuracy: 0.9823463687150839

Support Vector Machine Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      4103
           1       0.94      0.84      0.89       372

    accuracy                           0.98      4475
   macro avg       0.96      0.92      0.94      4475
weighted avg       0.98      0.98      0.98      4475
```

4 Kesimpulan

4.1 Confusion Matrix



4.2 Perbandingan Akurasi



4.3 Tingkat Kesalahan Klasifikasi

1. Logistic Regression:
   (21 + 66) / 4475 = 0.0194413407821229
   Sekitar 1.94 %
2. Decision Tree:
   (30 + 61) / 4475 = 0.0203351955307263
   Sekitar 2.03 %
3. K Nearest Neighbors:
   (19 + 67) / 4475 = 0.0192178770949721
   Sekitar 1.92 %
4. Naive Bayes:
   (160 + 63) / 4475 = 0.0498324022346369
   Sekitar 4.98 %
5. Random Forest:
   (29 + 62) / 4475 = 0.0203351955307263
   Sekitar 2.03 %
6. Support Vector Machine:
   (23 + 63) / 4475 = 0.0192178770949721
   Sekitar 1.92 %

5 Kontribusi Anggota

Project dibuat dengan membutuhkan waktu sekitar 12 jam kerja, memakan waktu kurang lebih 4-5 jam untuk mencari referensi, 5-6 jam untuk mengeksekusi coding dan paling banyak memakan waktu yaitu pada saat mencari parameter dengan GridSearch, 2-3 jam untuk membuat laporan.

1. Steven - 000.000.13433
   Exploratory Data Analysis, Modelling
2. Basilius Bias Astho Christyono - 000.000.13536
   Exploratory Data Analysis, Modelling, Menyusun Laporan
3. Raynaldi - 000.000.13654
   Exploratory Data Analysis, Menyusun Laporan

References

[1] Htru2 data set. https://archive.ics.uci.edu/ml/datasets/HTRU2. Accessed: 2015-03-18.