Traffic Congestion Prediction using Python

Bussiness Understanding

Peningkatan jumlah kendaraan tidak didukung dengan peningkatan kapasitas jalan dan kurangnya kesadaran akan kemacetan lalu lintas di Kota Bandung. Untuk menghilangkan kemacetan lalu lintas, berbagai langkah diterapkan untuk mengatur lalu lintas.

Banyak cara yang dapat dilakukan untuk mengatasi masalah kemacetan lalu lintas di Kota Bandung, diantaranya dengan penerapan teknologi.

Dari masalah di atas, saya memprediksi tingkat kemacetan di kota Bandung dengan menggunakan machine learning.

Data Understanding

Data yang disediakan oleh panitia DSLS 2023 merupakan data dari *crowdsourced/user generated* data program Waze for Cities dari Waze.

Terdapat 3 dataset, yaitu aggregate_median_jams, aggregate_median_irregularities, dan aggregate_alerts. Tabel-tabel tersebut merupakan hasil agregat dari *raw data* yang dirilis oleh Waze (jams, alerts, dan irregularities). Ketiga tabel agregat tersebut memiliki rentang waktu 6 Juli 2022 hingga 6 September 2022.

Tetapi pada prediksi yang saya lakukan, saya hanya menggunakan dataset *aggregate_median_irregularities* sesuai dengan tujuan yang saya tentukan, yaitu prediksi kemacetan di Kota Bandung.

Data Cleaning and Preprocesssing

Drop Features

Tahap ini hanya menghapus feature yang memiliki satu nilai dan feature-feature yang dirasa tidak akan memberikan informasi lebih.

```
[ ]  drop_column = ['kemendagri_kabupaten_kode', 'kemendagri_kabupaten_nama', 'cause_type', 'id', 'geometry']
     df.drop(drop_column, axis=1, inplace=True)
     df.head()
```

Handling Missing Values

	Features	Null Values	Percentage
0	cause_type	11281	100.00
1	street	2	0.02

Terdapat nilai null pada feature street dan cause_type. Karena feature cause_type sebelumnya sudah dihapus, maka akan kita hapus data pada feature street tersebut karena jumlahnya hanya sedikit.

```
[ ]  # Drop null values
     df.dropna(axis=0, inplace=True)

     # Make sure the value is gone
     df.isna().sum()
```

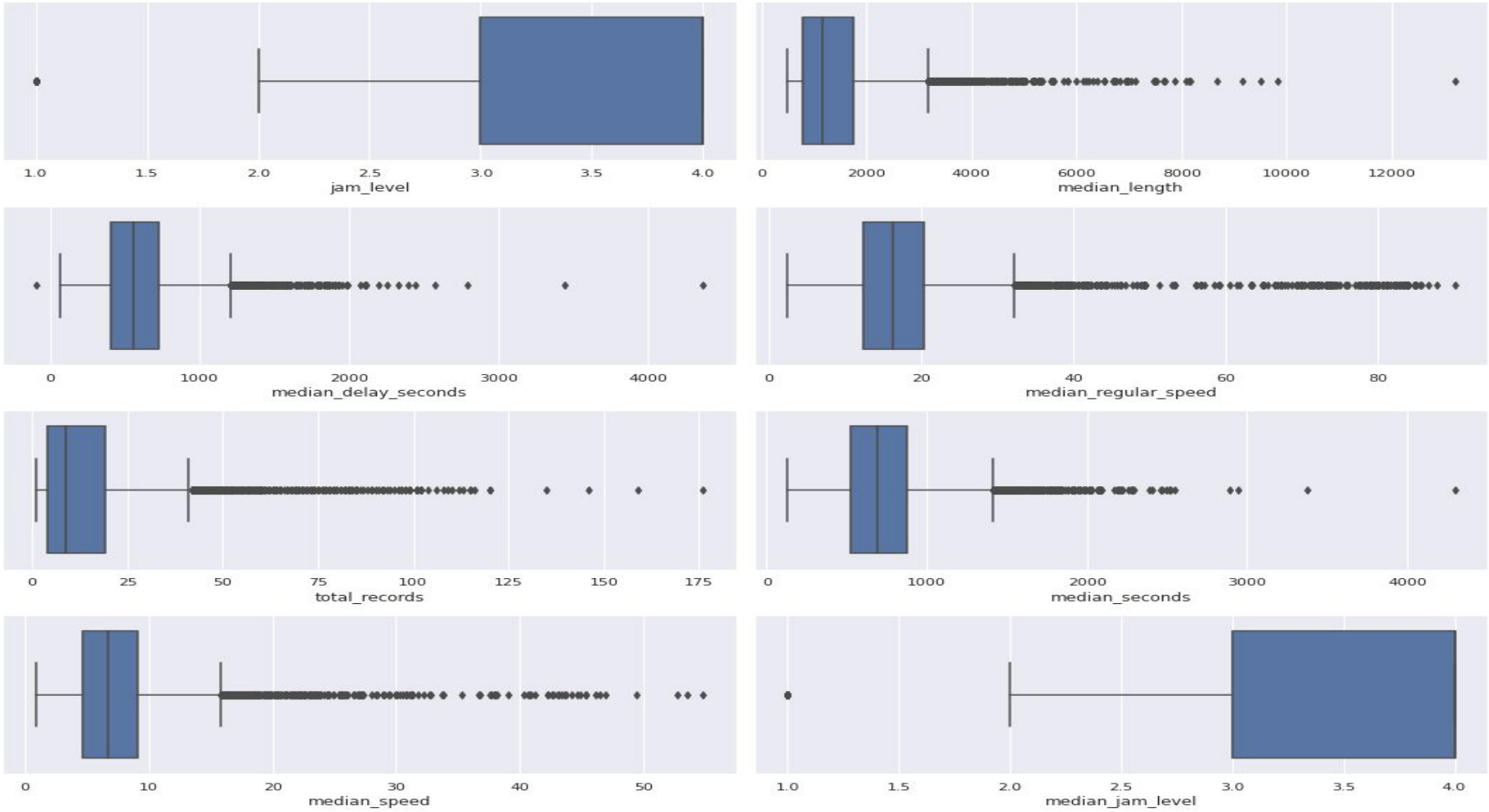
Converting Data Type

	feature	data_type	unique_sample
0	time	object	[2022-07-06 07:00:00.000, 2022-07-06 08:00:00....
1	street	object	[Terusan Buah Batu, Jenderal AH Nasution, N11 ...
2	jam_level	int64	[4, 3, 1, 2]
3	median_length	float64	[1922.0, 1819.0, 1064.0, 919.0]
4	median_delay_seconds	float64	[657.0, 421.0, 586.0, 558.5]
5	median_regular_speed	float64	[15.77, 17.939999, 14.52, 15.095]
6	total_records	int64	[13, 2, 11, 20]
7	median_seconds	float64	[844.0, 572.0, 675.0, 632.0]
8	median_speed	float64	[7.51, 11.6, 6.14, 4.97]
9	date	object	[2022-07-06, 2022-07-07, 2022-07-08, 2022-07-09]
10	median_jam_level	float64	[4.0, 3.0, 1.0, 2.0]

Pada tampilan diatas, terlihat feature **time** dan **date** belum bertipe data datetime, maka kita akan rubah.

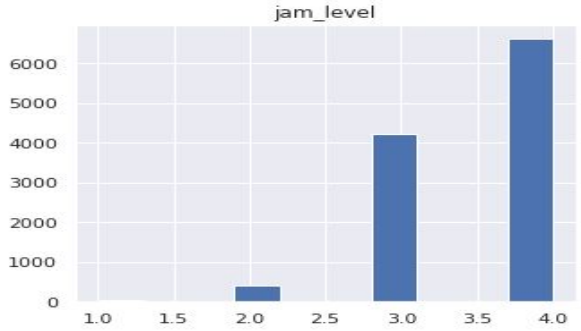
Exploratory Data Analysis

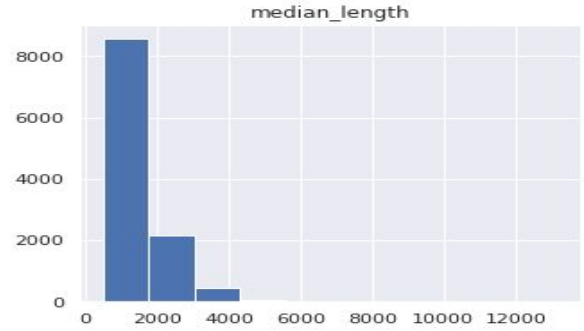
Data Outliers

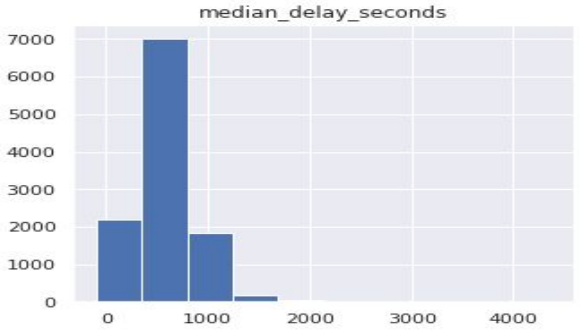


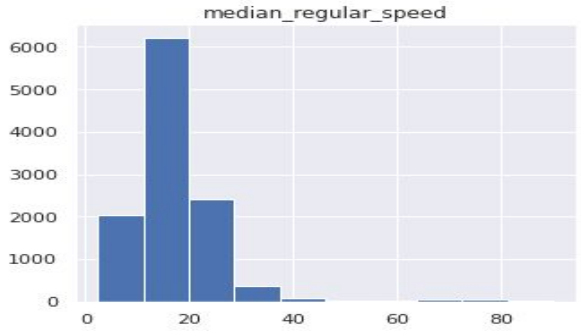
Pada boxplot diatas, terlihat bahwa hampir semua feature memiliki nilai outliers.

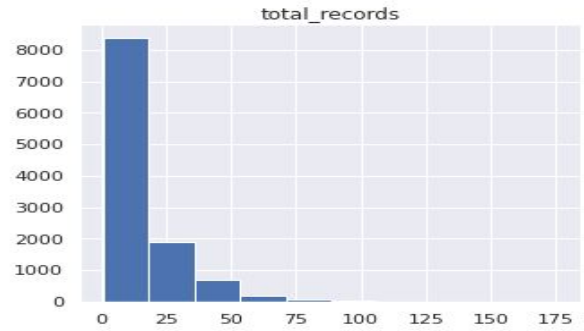
Data Distribution

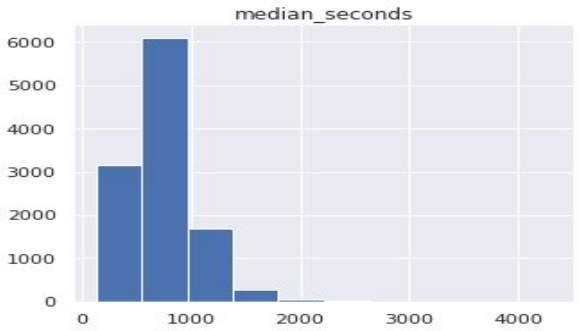


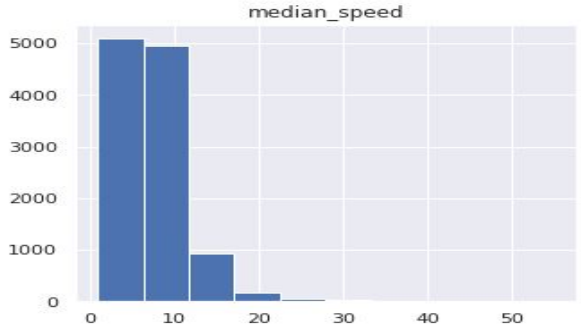


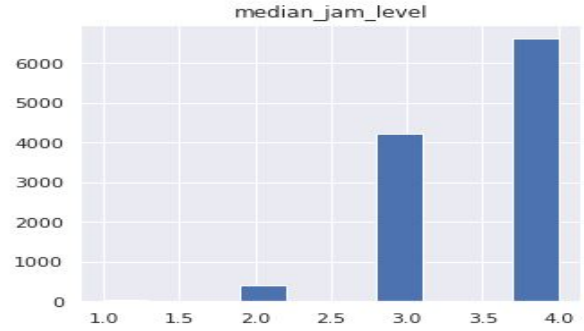






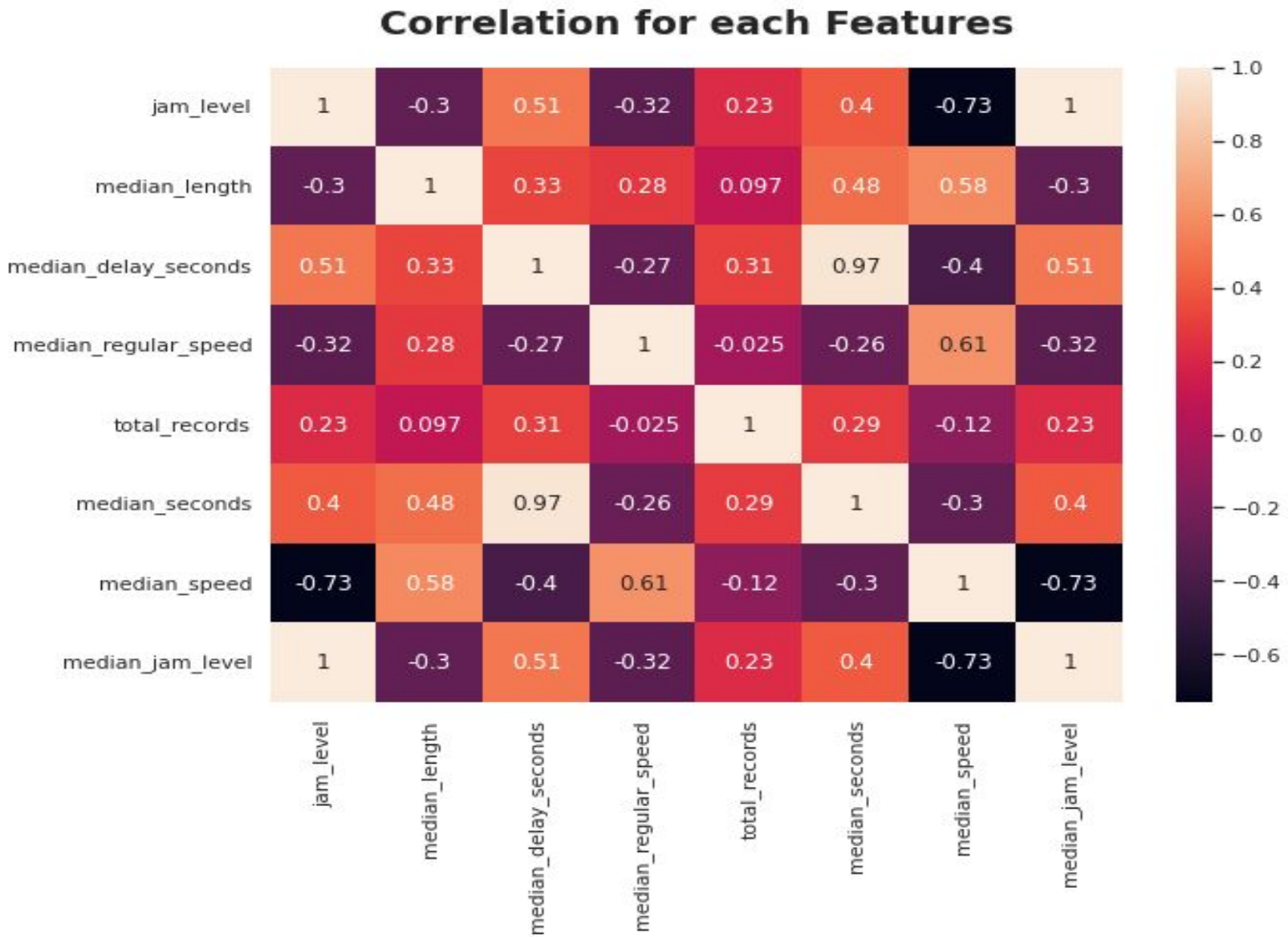






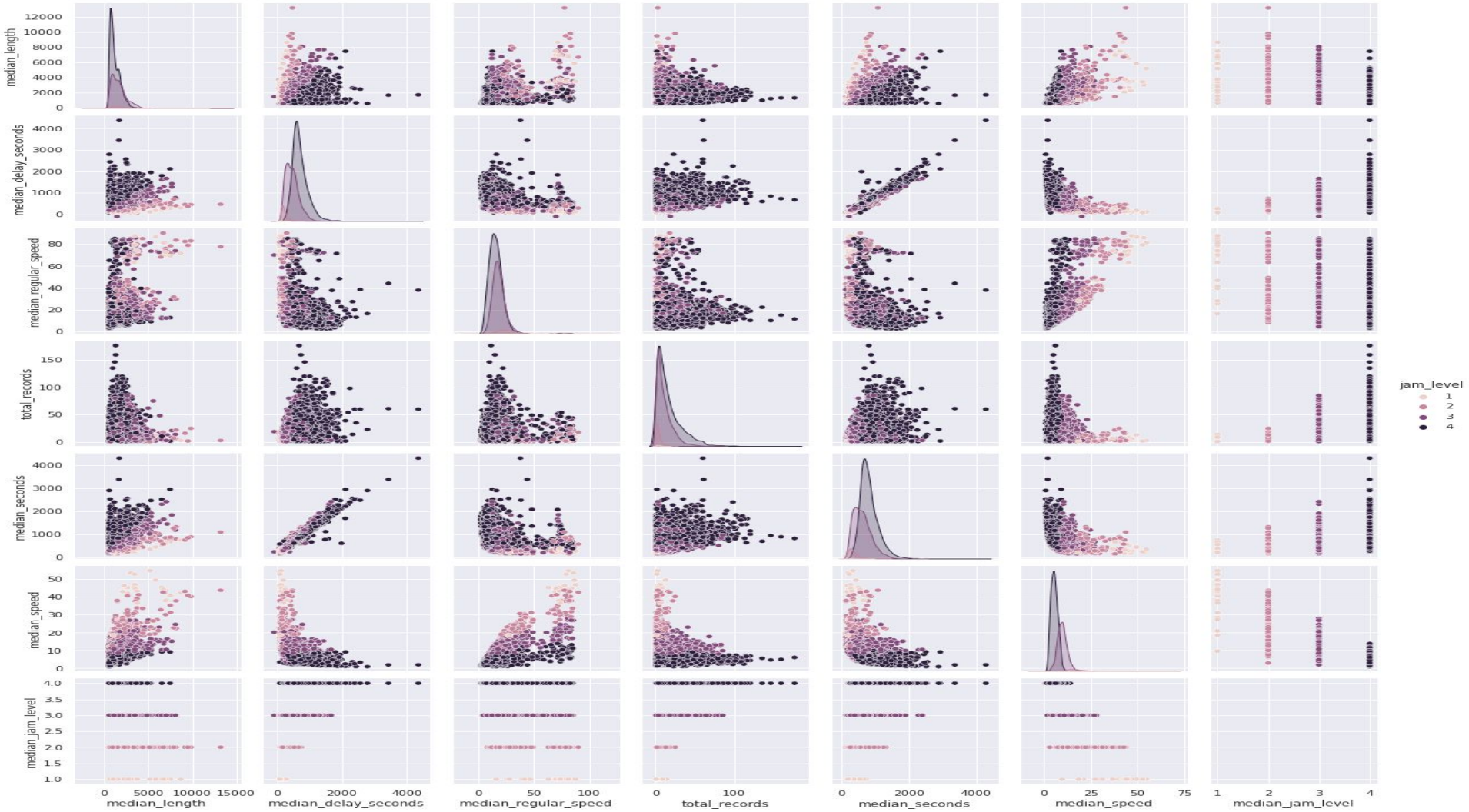
Pada boxplot diatas, terlihat bahwa beberapa tidak berdistribusi normal

Correlation Features



Terlihat bahwa ada beberapa feature yang memiliki korelasi positif dan negatif.

Pairplot


median_length
median_delay_seconds
median_regular_speed
total_records
median_seconds
median_speed
median_jam_level
jam_level
1
2
3
4

Feature Enginering

- Remove Features

Feature yang tidak dibutuhkan dihapus pada proses modelling, diantaranya : time, street, date, median_jam_level

- Feature Scaling

Karena data memiliki data outliers, maka digunakan Robust Scaling,

- OneHot Encoding

Karena data bertipe nominal, maka digunakan OneHot Encoding.

Resampling Dataset

- Train-Test Split

  Bagi data latih dan data uji dengan perbandingan 80:20

- Oversampling using SMOTE

  Karena variabel target terdapat imbalanced class, maka kita akan melakukan oversampling menggunakan metode SMOTE. Oversampling hanya dilakukan pada data training.

```
4    5965
3    3801
2     360
1      25
Name: jam_level, dtype: int64
```

```
4    5965
3    5965
2    5965
1    5965
Name: jam_level, dtype: int64
```

Modelling and Evaluation

Pemilihan Model

	score_mean	score_std
MLPClassifier	0.905931	0.003957
SVC	0.893519	0.005434
LogisticRegression	0.893430	0.003961
RandomForestClassifier	0.892278	0.005588
ExtraTreesClassifier	0.891125	0.004816
GradientBoostingClassifier	0.890238	0.003206
BaggingClassifier	0.883500	0.010652
XGBClassifier	0.881549	0.005235
KNeighborsClassifier	0.870024	0.008540
DecisionTreeClassifier	0.852203	0.005395
GaussianNB	0.840766	0.010455
AdaBoostClassifier	0.744213	0.032905

1. Membandingkan beberapa model machine learning yaitu KNN, Logistic Regresion, SVM, Naïve Bayer, MLP, Decision Tree, Random Forest, ExtraTrees, AdaBoost, GradientBoosting, Bagging, XGBoost.
2. Metode evaluasi menggunakan teknik cross validation.
3. Metrik evaluasi yang digunakan adalah accuracy, dan f-1 score karena semua label menjadi prioritas.
4. Pelatihan model dilakukan pada semua data

Fit and Evaluation

Selain dengan menggunakan Cross Validation, saya juga melakukan uji nilai akurasi pada data training dan testing yang bertujuan untuk melihat akurasi saat data training dan testing dan perbedaaannya. Maka didapat hasil sebagai berikut :

	train score	test score	difference
GaussianNB	0.786588	0.785461	0.001127
AdaBoostClassifier	0.692456	0.695922	0.003466
SVC	0.895725	0.867908	0.027817
LogisticRegression	0.908592	0.874113	0.034478
XGBClassifier	0.924811	0.869681	0.055131
GradientBoostingClassifier	0.942330	0.883865	0.058465
MLPClassifier	0.952523	0.893972	0.058551
ExtraTreesClassifier	1.000000	0.903723	0.096277
RandomForestClassifier	0.999975	0.900709	0.099266
KNeighborsClassifier	0.958382	0.852837	0.105545
BaggingClassifier	0.995809	0.883865	0.111944
DecisionTreeClassifier	1.000000	0.847163	0.152837

- Terlihat bahwa GaussianNB memiliki selisih nilai akurasi yang paling kecil yaitu hanya 0,1%. Walaupun selisih nilai GaussianNB paling kecil, namun nilai akurasi data latih dan uji lebih rendah dibandingkan model lainnya.

- Pada keseluruhan proses di atas, saya memilih model SVC karena memiliki nilai akurasi yang cukup tinggi pada data training dan test, dan selisihnya tidak terlalu banyak, hanya 2%. Meskipun cross-validated MLPC classifier memiliki akurasi yang tinggi, namun membutuhkan waktu yang lebih lama untuk melatih model dibandingkan SVC, meskipun keduanya memiliki nilai akurasi yang hampir sama.

Hyperparameter Tuning

Pada tahap ini kita melakukan hyperparameter pada model SVM. Setelah itu kita bandingkan performa hasil hyperparameter dengan default parameter.

	train_acc	test_acc	f1-score
svm_default	89.57	86.79	79.18
svm_tuned	95.58	89.01	79.18

Dari hasil disamping, hyperparameter tuning meningkatkan akurasi pada data training dan data testing tetapi memiliki perbedaan persentase yang cukup besar. Maka dapat dikatakan bahwa bahwa model mengalami sedikit overfitting. Maka pada training model ini saya akan memilih model SVM dengan default parameter karena perbedaan akurasi pada data training dan testing tidak terlalu jauh.

Evaluation

	precision	recall	f1-score	support
1	0.57	1.00	0.73	4
2	0.60	0.84	0.70	45
3	0.81	0.85	0.83	410
4	0.94	0.88	0.91	669
accuracy			0.87	1128
macro avg	0.73	0.89	0.79	1128
weighted avg	0.88	0.87	0.87	1128

TERIMA KASIH