Rakamin Academy

Shipment Arrival Prediction

Dokumen Laporan Final Project

(dipresentasikan setiap sesi mentoring)

Latar Belakang Masalah

Masalah-masalah yang ada:

- 59.67% paket terlambat sampai

Pertanyaan: Bagaimana cara memprediksi paket yang datang terlambat?

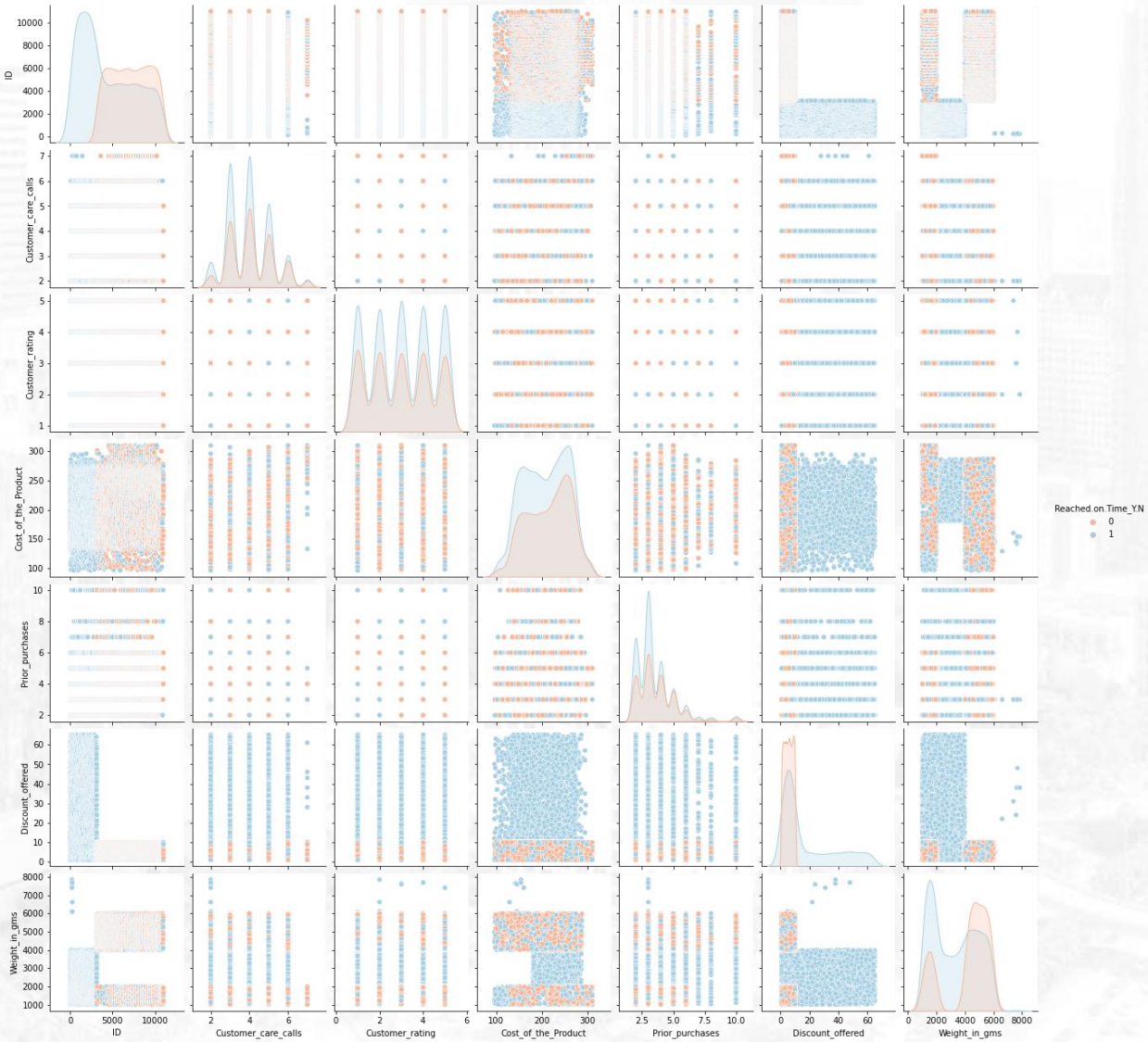
Solusi: Membuat model machine learning untuk memprediksi paket akan datang terlambat atau tidak.

Penjelasan: Model ini selanjutnya dapat dipakai untuk membantu membuat tools untuk memberikan notifikasi keterlambatan kepada pelanggan agar pelanggan tidak terlalu banyak menelepon ke CS dan membuat pekerjaan CS membludak.

EDA

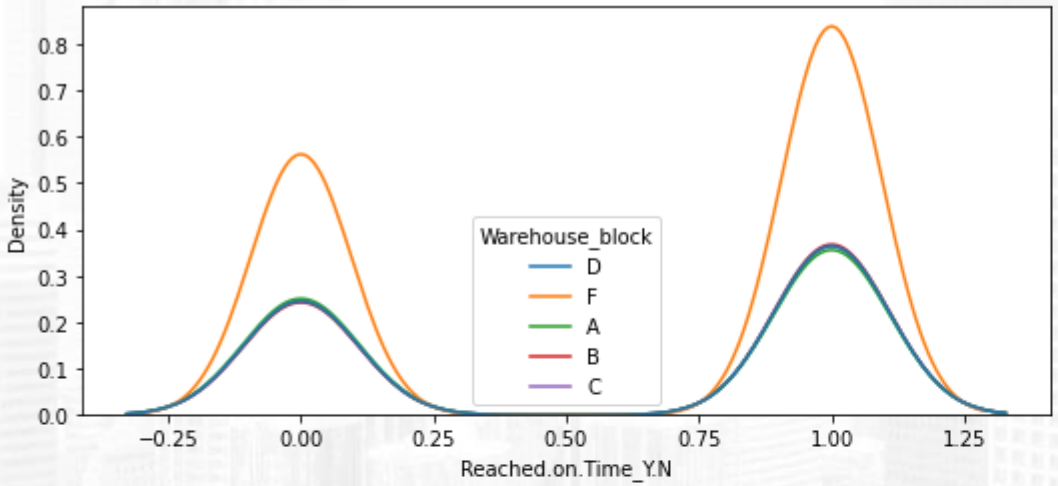
Rakamin Academy

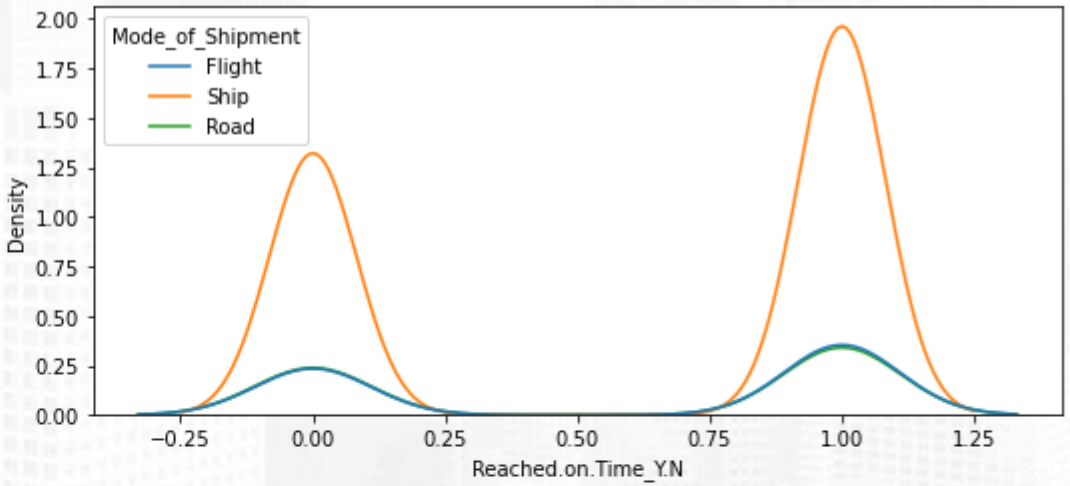
DATA VISUALISASI (feature numerikal vs target)

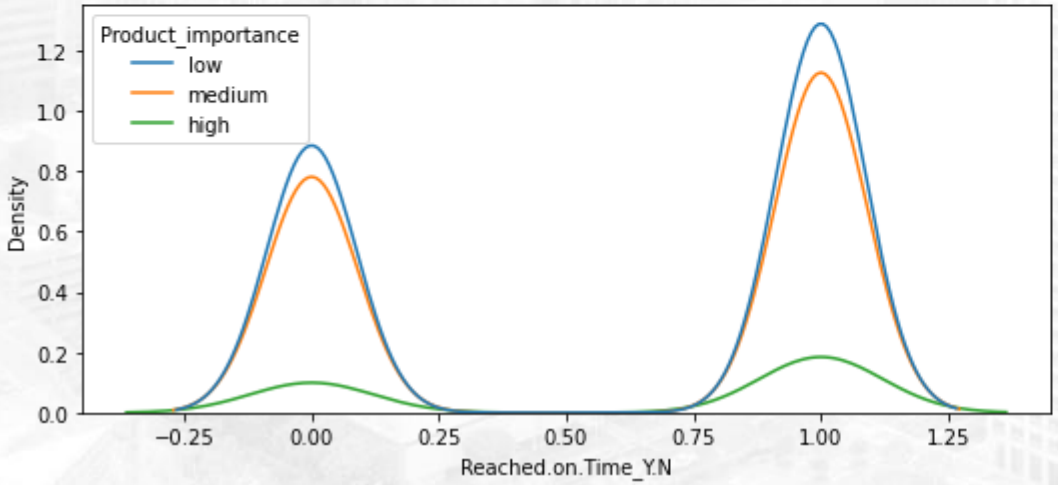


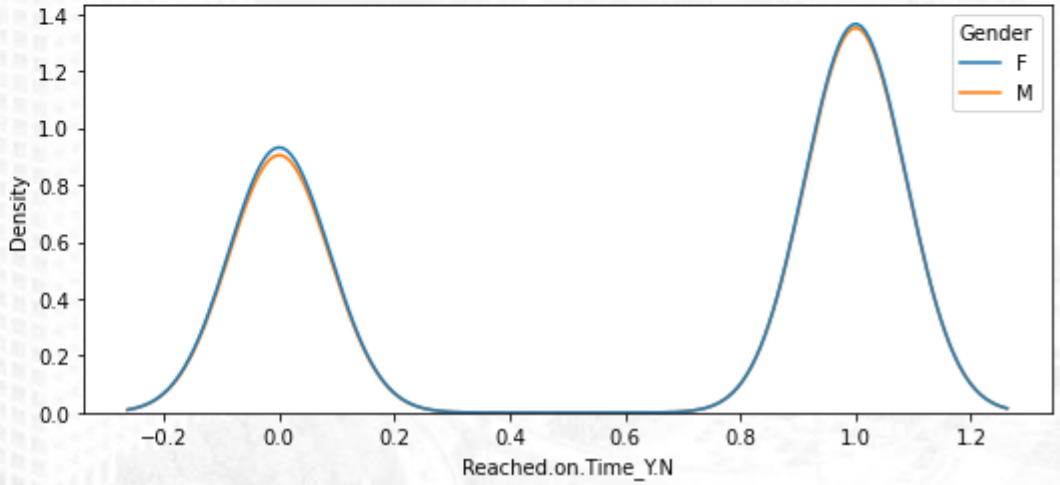
EDA

DATA VISUALISASI (feature kategorikal vs target)









EDA

Insight yang dapat diambil dari visualisasi feature vs target:

- `**Customer_care_calls**`: Mayoritas pelanggan akan menelepon CS sebanyak 3-4x terlepas produk terlambat atau tidak. 94.2% pelanggan yang menelpon ke CS lebih dari 2x.
- `**Customer_rating**`: Pelanggan dengan produk tepat waktu paling banyak memberikan rating 1 dan pelanggan terlambat paling banyak memberikan rating 3. 60.36% pelanggan memberikan rating di bawah 4
- `**Prior_purchases**`: Mayoritas pelanggan telah melakukan pembelian sebanyak 3x sebelumnya, terlepas produknya terlambat atau tidak.
- `**Cost_of_the_Product**`: Produk yang terlambat memiliki median harga yang lebih rendah dibandingkan dengan produk yang tepat waktu.
- `**Discount_offered**`: Produk yang diberikan diskon besar cenderung datang terlambat dan produk yang diberikan diskon < $10 cenderung datang tepat waktu.
- `**Weight_in_gms**`: Produk yang memiliki berat 4000-5500 gram cenderung datang tepat waktu, sedangkan produk yang datang terlambat cenderung memiliki berat yang lebih ringan. IQR berat produk tepat waktu lebih sempit dibandingkan IQR berat produk terlambat.
- `**Warehouse_block**`: Mayoritas produk dikirim dari Blok F, terlepas produk datang terlambat atau tidak.
- `**Mode_of_Shipment**`: Mayoritas produk dikirim menggunakan kapal, terlepas produk datang terlambat atau tidak.
- `**Product_importance**`: Mayoritas produk memiliki prioritas rendah, terlepas produk datang terlambat atau tidak.
- `**Gender**`: Mayoritas produk datang terlambat, terlepas dari jenis kelamin pelanggan.

Data Cleansing (Null and duplicate values)

Pertama kita cek apakah ada data yang kosong dan duplicate.

- Tidak ada data null.
- Tidak ada data duplicate.

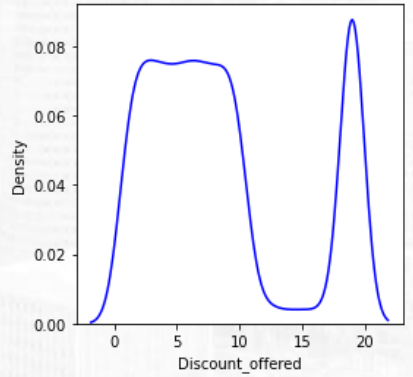
```
[26] df.isna().sum()

ID                     0
Warehouse_block        0
Mode_of_Shipment       0
Customer_care_calls    0
Customer_rating        0
Cost_of_the_Product    0
Prior_purchases        0
Product_importance     0
Gender                 0
Discount_offered       0
Weight_in_gms          0
Reached.on.Time_Y.N    0
dtype: int64
```

```
[27] df.duplicated().any()

False
```

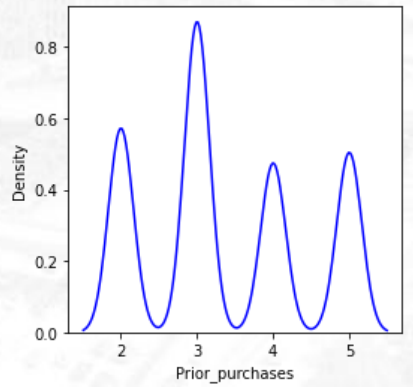
Data Cleansing (Outlier)

- Kami memutuskan untuk mengganti nilai outlier dari kolom `**Discount_offered**` dan `**Prior_purchases**` menjadi **nilai 1,5(IQR)** masing-masing karena memang nilai outliernya cukup banyak dan besar yang apabila dibuang akan mengurangi informasi dari data di sini.

```
Q1 = df['Discount_offered'].quantile(0.25)
Q3 = df['Discount_offered'].quantile(0.75)
IQR = Q3 - Q1
low_limit = Q1 - (1.5 * IQR)
high_limit = Q3 + (1.5 * IQR)

df['Discount_offered'] = np.where(df['Discount_offered'] > high_limit, high_limit, df['Discount_offered'])
plt.figure(figsize=(4,4))
sns.kdeplot(x=df['Discount_offered'],color='blue')
```

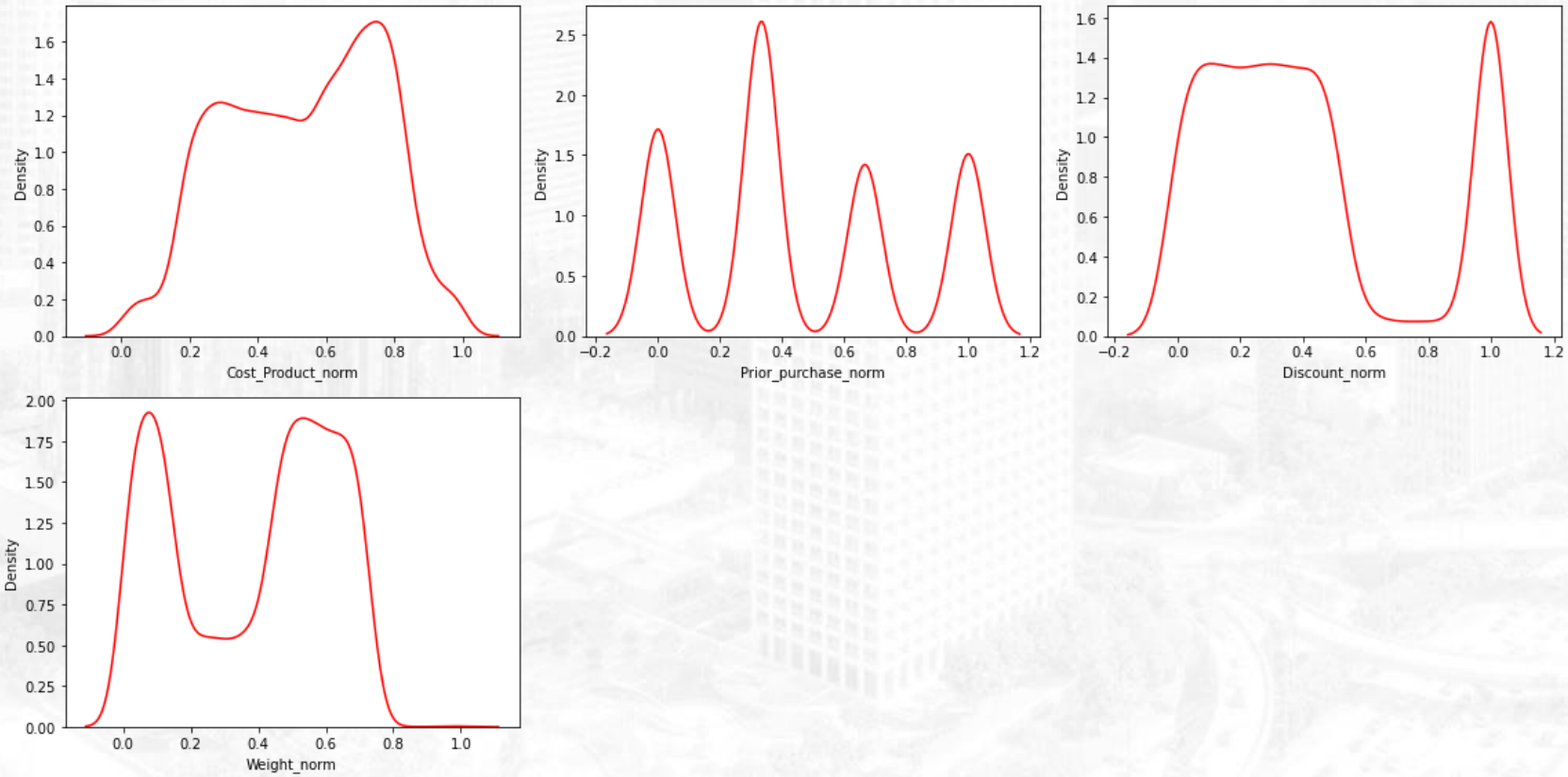



```
Q1 = df['Prior_purchases'].quantile(0.25)
Q3 = df['Prior_purchases'].quantile(0.75)
IQR = Q3 - Q1
low_limit = Q1 - (1.5 * IQR)
high_limit = Q3 + (1.5 * IQR)

df['Prior_purchases'] = np.where(df['Prior_purchases'] > high_limit, math.floor(high_limit), df['Prior_purchases'])
plt.figure(figsize=(4,4))
sns.kdeplot(x=df['Prior_purchases'],color='blue')
```



Feature Transformation (Normalization)

- Normalisasi digunakan ke feature `**Cost_of_the_Product**`, `**Discount_offered**`, `**Prior_purchase**`, dan `**Weight_in_gms**`



Class Imbalance

MENCARI TARGET FEATURE YANG DATANYA TIDAK BERIMBANG

Rasio nilai 0 dan 1 pada kolom target adalah 40:60 dan distribusinya tidak sangat timpang, maka dari itu tidak ada class imbalance pada data ini.

```
late = df.groupby(['Reached.on.Time_Y.N']).agg({'ID':'nunique'}).reset_index()
late.columns = ['Late','Total']
late['Percentage'] = late.apply(lambda x: round(x['Total']/10999*100,2),axis=1)
late
```

	Late	Total	Percentage
0	0	4436	40.33
1	1	6563	59.67

Feature Encoding

1. Warehouse F kami ganti ke Warehouse E

```
# ganti warehouse F ke E
df['Warehouse_block'] = np.where(df['Warehouse_block']=='F', 'E', df['Warehouse_block'])
df['Warehouse_block'].value_counts()
```

```
E    3666
D    1834
A    1833
B    1833
C    1833
Name: Warehouse_block, dtype: int64
```

2. Untuk kolom Product Importance dan Gender kami menggunakan proses perubahan data manual

```
# label encoding mapping cats yang punya 2 distinct value / ordinal
# Product_importance dan Gender
mapping_gender = {
    'F' : 0,
    'M' : 1
}

mapping_product_importance = {
    'low' : 0,
    'medium' : 1,
    'high' : 2
}

df['Gender'] = df['Gender'].map(mapping_gender)
df['Product_importance'] = df['Product_importance'].map(mapping_product_importance)
```

3. Untuk feature Warehouse Block dan Mode of Shipment, kami menggunakan One Hots Encoding, untuk melihat tingkatan antar data lebih representatif :

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ID                      10999 non-null  int64
 1   Customer_care_calls     10999 non-null  int64
 2   Customer_rating         10999 non-null  int64
 3   Product_importance      10999 non-null  int64
 4   Gender                  10999 non-null  int64
 5   Reached.on.Time_Y.N     10999 non-null  int64
 6   Cost_Product_norm       10999 non-null  float64
 7   Prior_purchase_norm     10999 non-null  float64
 8   Discount_norm           10999 non-null  float64
 9   Weight_norm             10999 non-null  float64
 10  Warehouse_block_A       10999 non-null  uint8
 11  Warehouse_block_B       10999 non-null  uint8
 12  Warehouse_block_C       10999 non-null  uint8
 13  Warehouse_block_D       10999 non-null  uint8
 14  Warehouse_block_E       10999 non-null  uint8
 15  Mode_of_Shipment_Flight 10999 non-null  uint8
 16  Mode_of_Shipment_Road   10999 non-null  uint8
 17  Mode_of_Shipment_Ship   10999 non-null  uint8
dtypes: float64(4), int64(6), uint8(8)
```

Feature Selection & Feature Extraction

Feature yang tidak diseleksi, yakni `**ID**`, `**Customer_care_calls**`, `**Customer_rating**`

Hal ini disebabkan karena ketiga feature tersebut tidak berkorelasi terkait tepat waktu atau tidaknya pengiriman.

Feature 2 dan 3 dibuang karena dapat menyebabkan data leakage, yang berarti bahwa data dari kedua feature ini sebenarnya belum ada di saat model digunakan untuk memprediksi pengiriman akan terlambat atau tidak.

```
df = df.drop(columns = ['ID','Customer_care_calls','Customer_rating'])
```

Feature extraction:

Kami tidak menambahkan fitur baru untuk data ini

4 Feature Tambahan

1. Waktu Pengiriman
Mengetahui kapan waktu yang sibuk dan longgar di setiap warehouse, sehingga dapat digunakan sebagai alat antisipasi potensi keterlambatan pengiriman.
2. Estimasi Waktu Barang Tiba
Mengetahui durasi pengiriman barang dari warehouse menuju tempat pengiriman, sehingga dapat membantu model menganalisis tingkat efektifitas dan efisiensi suatu pengiriman.
3. Region (Domestic / International)
Dokumen dan persyaratan pengiriman internasional memerlukan waktu yang lebih lama, sehingga dapat meningkatkan permodelan yang lebih akurat apabila pengiriman dapat dikategorikan berdasarkan region.
4. Jarak
Semakin jauh jarak yang ditempuh dalam pengiriman, menyebabkan lamanya waktu pengiriman, dengan adanya fitur jarak diharapkan model dapat memprediksi keterlambatan dari suatu pengiriman.

Split Training & Testing Data

```
X = df.drop(columns=['Reached.on.Time_Y.N','Unnamed: 0','prior_purchase_std','product_cost_norm','discount_norm','weight_norm'])
y = df[['Reached.on.Time_Y.N']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42,stratify=y)
```

Split data training dan testing data menggunakan rasio 70 : 30 dari jumlah 10,999 data, dengan distribusi sebagai berikut:

- data training sejumlah 7,699 data
- data testing sejumlah 3,300 data

Pemilihan Model

Algoritma yang kami pakai dalam tugas ini adalah:

1. Logistic Regression
2. K-Nearest Neighbor
3. Decision Tree
4. Random Forest
5. Gradient Boosting
6. Gaussian Naive-Bayes
7. AdaBoost
8. XGBoost

Logistic Regression

```
Score:  0.6376
Accuracy (Test Set):  0.6376
Precision (Test Set):  0.6924
Recall (Test Set):  0.7064
F1-Score (Test Set):  0.6993
roc-auc (test-proba):  0.7208
roc-auc (train-proba):  0.7209
```

```
XGboost

Score:  0.6594
Accuracy (Test Set):  0.6594
Precision (Test Set):  0.7413
Recall (Test Set):  0.6592
F1-Score (Test Set):  0.6978
roc-auc (test-proba):  0.746
roc-auc (train-proba):  0.9724
```

```
K-Nearest Neighbors

Score:  0.6209
Accuracy (Test Set):  0.6209
Precision (Test Set):  0.6945
Recall (Test Set):  0.6511
F1-Score (Test Set):  0.6721
roc-auc (test-proba):  0.6865
roc-auc (train-proba):  0.8584

==============================

Decision Tree

Score:  0.6476
Accuracy (Test Set):  0.6476
Precision (Test Set):  0.7069
Recall (Test Set):  0.6993
F1-Score (Test Set):  0.7031
roc-auc (test-proba):  0.6352
roc-auc (train-proba):  1.0

==============================

Random Forest

Score:  0.6585
Accuracy (Test Set):  0.6585
Precision (Test Set):  0.7425
Recall (Test Set):  0.6546
F1-Score (Test Set):  0.6958
roc-auc (test-proba):  0.7336
roc-auc (train-proba):  1.0
```

```
Gradient Boosting

Score:  0.6761
Accuracy (Test Set):  0.6761
Precision (Test Set):  0.9061
Recall (Test Set):  0.5099
F1-Score (Test Set):  0.6526
roc-auc (test-proba):  0.7353
roc-auc (train-proba):  0.8154

==============================

Gaussian Naive Bayes

Score:  0.6558
Accuracy (Test Set):  0.6558
Precision (Test Set):  0.8069
Recall (Test Set):  0.5561
F1-Score (Test Set):  0.6584
roc-auc (test-proba):  0.7262
roc-auc (train-proba):  0.726

==============================

AdaBoost

Score:  0.677
Accuracy (Test Set):  0.677
Precision (Test Set):  0.8503
Recall (Test Set):  0.5566
F1-Score (Test Set):  0.6728
roc-auc (test-proba):  0.7454
roc-auc (train-proba):  0.7611
```

Hyperparameter Tuning

Logistic Regression

```
Score:  0.7201
Accuracy (Test Set):  0.6361
Precision (Test Set):  0.6881
Recall (Test Set):  0.7136
F1-Score (Test Set):  0.7006
roc-auc (test-proba):  0.7201
roc-auc (train-proba):  0.7203

Best penalty : l2
Best C : 0.1
Best solver : saga
```

```
K-Nearest Neighbors tuned

Score:  0.7047
Accuracy (Test Set):  0.637
Precision (Test Set):  0.7194
Recall (Test Set):  0.642
F1-Score (Test Set):  0.6785
roc-auc (test-proba):  0.7047
roc-auc (train-proba):  0.7919

==========

Decision Tree tuned

Score:  0.7123
Accuracy (Test Set):  0.6394
Precision (Test Set):  0.743
Recall (Test Set):  0.6049
F1-Score (Test Set):  0.6669
roc-auc (test-proba):  0.7123
roc-auc (train-proba):  0.7803

==========

Random Forest tuned

Score:  0.7315
Accuracy (Test Set):  0.6645
Precision (Test Set):  0.7968
Recall (Test Set):  0.5876
F1-Score (Test Set):  0.6764
roc-auc (test-proba):  0.7315
roc-auc (train-proba):  0.8391
```

```
Gradient Boosting tuned

Score:  0.742
Accuracy (Test Set):  0.6685
Precision (Test Set):  0.776
Recall (Test Set):  0.6247
F1-Score (Test Set):  0.6922
roc-auc (test-proba):  0.742
roc-auc (train-proba):  0.7488

============================================================

Fitting 5 folds for each of 10 candidates, totalling 50 fits
Gaussian Naive Bayes tuned

Score:  0.7262
Accuracy (Test Set):  0.6558
Precision (Test Set):  0.8069
Recall (Test Set):  0.5561
F1-Score (Test Set):  0.6584
roc-auc (test-proba):  0.7262
roc-auc (train-proba):  0.726

============================================================

XGboost tuned

Score:  0.7341
Accuracy (Test Set):  0.6515
Precision (Test Set):  0.731
Recall (Test Set):  0.6582
F1-Score (Test Set):  0.6927
roc-auc (test-proba):  0.7341
roc-auc (train-proba):  0.9802
```

Evaluasi Model (Kesimpulan)

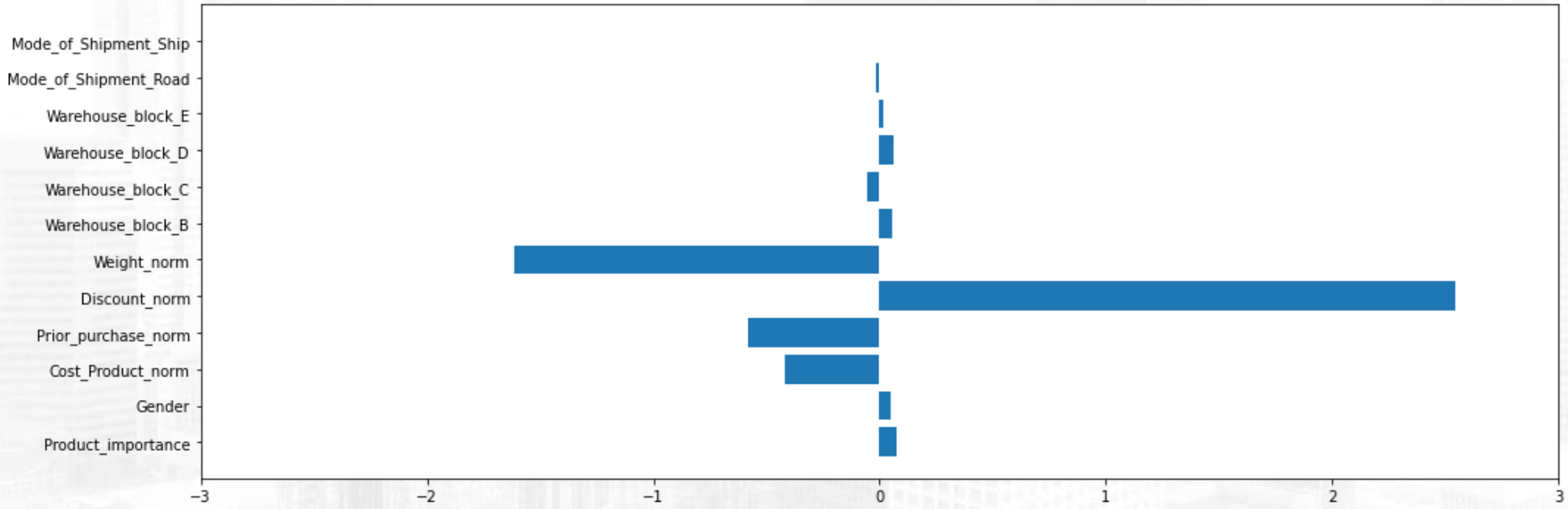
Dari model-model yang telah kami coba, belum ada model yang memiliki semua dari:

1. Nilai AUC-train dan AUC-test yang berbeda tipis,
2. Nilai AUC-train < 1.00, dan
3. Nilai Recall yang besar.

Dengan kata lain, model-model yang telah kami coba masih overfitting.

Model Logistic Regression:
- Score: 0.7201
- Accuracy (Test Set): 0.6361
- Precision (Test Set): 0.6881
- **Recall (Test Set): 0.7136**
- F1-Score (Test Set): 0.7006
- roc-auc (test-proba): 0.7201
- roc-auc (train-proba): 0.7203

Feature Importance



- `**Discount_norm**` memiliki score sebesar 2.540145
- `**Weight_norm**` memiliki score sebesar -1.616171
- `**Prior_purchase_norm**` memiliki score sebesar -0.580930
- `**Cost_Product_norm**` memiliki score sebesar -0.420324

Bila koefisien regresi memiliki nilai positif, berarti semakin besar nilai fitur tersebut, semakin tinggi kemungkinan terlambat.
Bila koefisien regresi memiliki nilai negatif, berarti semakin besar nilai fitur tersebut, semakin tinggi kemungkinan tepat waktu.