cakap

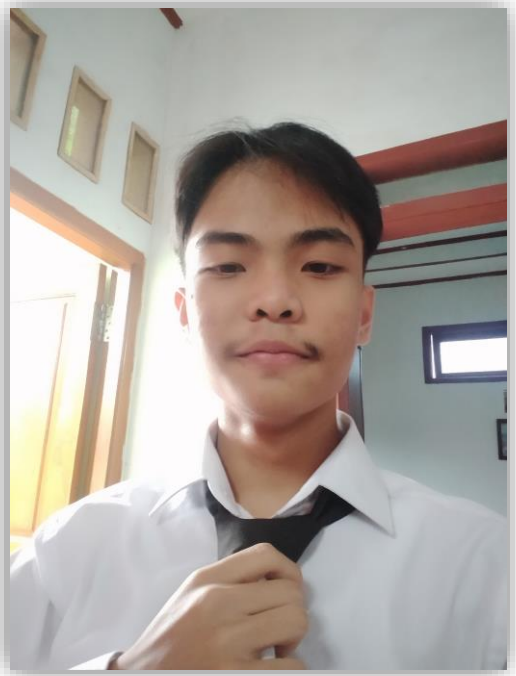
Data Scientist Capstone Project

Introduction

Identitas diri

Submission Proyek Akhir
Studi Independen Bersertifikat Batch 6: Data Scientist

- Nama : Roni Antonius Sinabutar
- Kelas : Data Science A
- Email : aantoniusron@gmail.com
- Universitas : Universitas Pendidikan Indonesia
- Tempat Tinggal : Cianjur, Jawa Barat



Daftar Isi

Problem Overview

Problems

Hampir seluruh aktivitas pada perusahaan produksi pakaian masih dilakukan secara manual, sehingga akan cukup sulit dan membutuhkan waktu dan proses yang lama jika ingin melakukan intervensi jika efektivitas kinerja karyawan mulai menurun dan juga ketika ingin memberikan insentif jika kinerjanya baik.

Goals

Membuat model yang mampu memprediksi secara langsung mengenai seberapa efektif kinerja karyawan melalui produktivitas aktual.

Data Collection & Data understanding

Data Collection

Data Collection merupakan suatu upaya untuk mengumpulkan data baik dari internal maupun eksternal. Pada projek akhir yang saya namakan **'Prediksi Produktivitas Buruh Pada Perusahaan Pakaian'** akan mengambil atau mengumpulkan data dari eksternal, lalu kemudian mengaksesnya dengan `read_csv`.

Prediksi Produktivitas Buruh Pada Perusahaan Pakaian [Regresi]

In [1]:
```
import warnings
import os
warnings.filterwarnings('ignore')
```

In [2]:
```
from sklearn.model_selection       import train_test_split, GridSearchCV, RandomizedSearchCV, cross_val_score
from sklearn.neural_network        import MLPRegressor
from sklearn.preprocessing         import MinMaxScaler, StandardScaler, LabelEncoder, OrdinalEncoder, OneHotEncoder, RobustS
from sklearn.linear_model          import LinearRegression
from sklearn.neighbors             import KNeighborsRegressor
from sklearn.ensemble              import RandomForestRegressor
from sklearn.metrics               import mean_squared_error, mean_absolute_error, r2_score
from sklearn.tree                  import DecisionTreeRegressor
from sklearn.svm                   import SVR
from pyforest                      import *
```

In [3]:
```
df = pd.read_csv('C:/Users/ASUS/kode/Projek/9. Produktifitas_Buruh/garments_worker_productivity.csv')
```

Data Understanding

Tahapan yang melibatkan eksplorasi awal data untuk memahami karakteristiknya.

- Date : Tanggal
- Day : Hari dalam seminggu
- Quarter : Kuartal dalam setahun
- Department : Nama departemen
- team_no : Nomor tim
- no_of_workers : Jumlah pekerja dalam tim
- no_of_style_change : Jumlah perubahan desain produk
- targeted_productivity : Target produktivitas tim per hari
- Smv : Waktu standar untuk menyelesaikan tugas
- Wip : Jumlah produk yang belum selesai diproduksi
- over_time : Waktu tambahan yang digunakan oleh tim
- Incentive : Insentif yang diberikan kepada pekerja
- idle_time : Waktu idle karena gangguan produksi
- idle_men : Jumlah pekerja yang idle karena gangguan produksi
- actual_productivity : Tingkat produktivitas pekerja (0-1)

Memahami Data

Struktur Data

Struktur Data

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1197 entries, 0 to 1196
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   date                   1197 non-null   object 
 1   quarter                1197 non-null   object 
 2   department             1197 non-null   object 
 3   day                    1197 non-null   object 
 4   team                   1197 non-null   int64  
 5   targeted_productivity  1197 non-null   float64
 6   smv                    1197 non-null   float64
 7   wip                    691 non-null    float64
 8   over_time              1197 non-null   int64  
 9   incentive              1197 non-null   int64  
 10  idle_time              1197 non-null   float64
 11  idle_men               1197 non-null   int64  
 12  no_of_style_change     1197 non-null   int64  
 13  no_of_workers          1197 non-null   float64
 14  actual_productivity    1197 non-null   float64
dtypes: float64(6), int64(5), object(4)
memory usage: 140.4+ KB
```

Data Hilang

Missing value

```
1 df.isnull().sum()
```

```
date                       0
quarter                    0
department                 0
day                        0
team                       0
targeted_productivity      0
smv                        0
wip                      506
over_time                  0
incentive                  0
idle_time                  0
idle_men                   0
no_of_style_change         0
no_of_workers              0
actual_productivity        0
dtype: int64
```

Dimensi Data

Dimensi data

```
1 df.shape
```

```
(1197, 15)
```

11197 record data serta 15 variabel

Memahami Data

Imbalance Case

Imbalance Case terhadap variabel target

```
1 df3['actual_productivity'].value_counts()
```

```
actual_productivity
0.800402    24
0.971867    12
0.850137    12
0.750651    11
0.850502    11
            ..
0.800034     1
0.800024     1
0.769293     1
0.750031     1
0.394722     1
Name: count, Length: 879, dtype: int64
```

Variabel target merupakan variabel **kontinyu** sehingga akan sangat cocok untuk diterapkan regresi yaitu tugas prediksi.

Data Cleaning

Handling Missing Value

Problem

Missing value

```
1 df.isnull().sum()
```

```
date                       0
quarter                    0
department                 0
day                        0
team                       0
targeted_productivity      0
smv                        0
wip                      506
over_time                  0
incentive                  0
idle_time                  0
idle_men                   0
no_of_style_change         0
no_of_workers              0
actual_productivity        0
dtype: int64
```

Solusi

```
1 df['wip'].fillna(df['wip'].mean(), inplace=True)
```

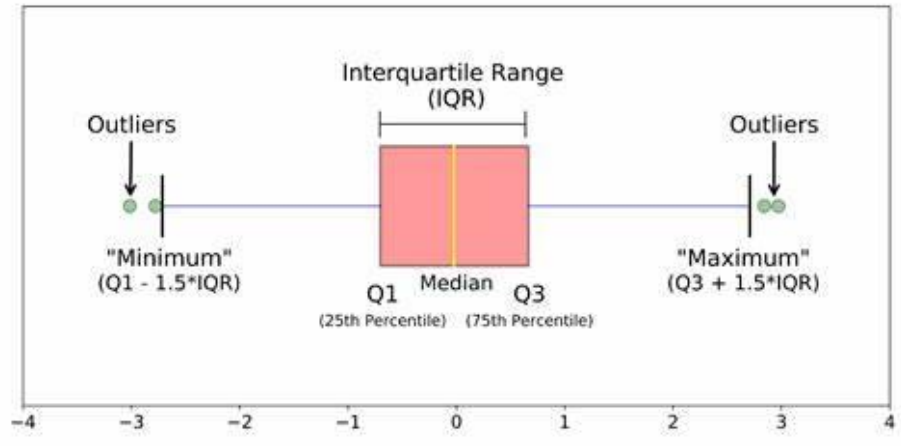
```
1 df.isnull().sum()
```

```
date                       0
quarter                    0
department                 0
day                        0
team                       0
targeted_productivity      0
smv                        0
wip                        0
over_time                  0
incentive                  0
idle_time                  0
idle_men                   0
no_of_style_change         0
no_of_workers              0
actual_productivity        0
dtype: int64
```

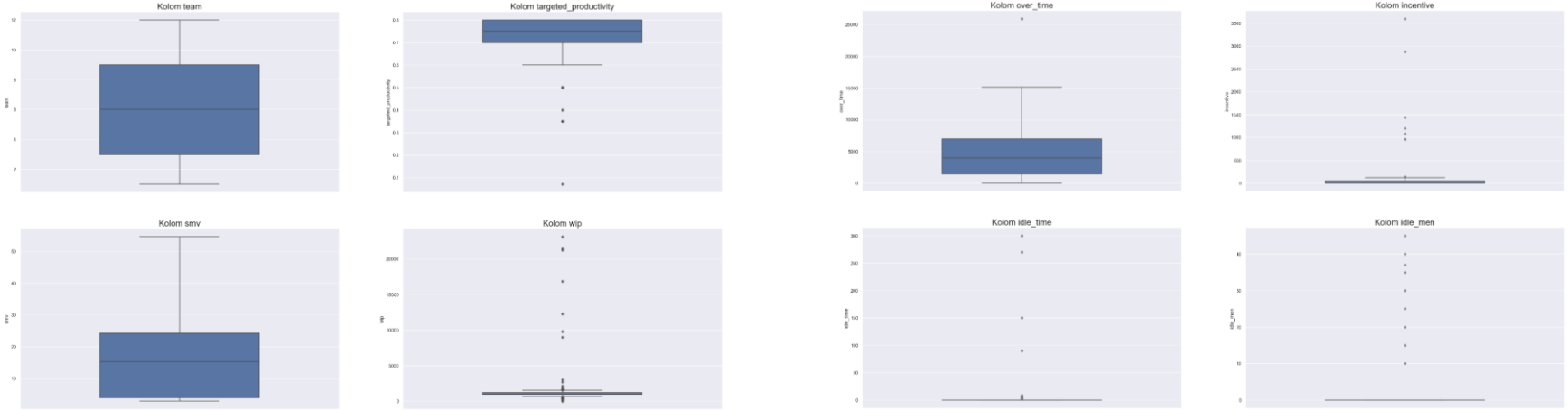
Solusi untuk 500 record data yang mengalami missing value tersebut tidak bisa aku isi dengan 0 atau menghapus kolomnya, karena 50% data mengalami missing value. Solusinya mungkin bisa diisi dengan mean atau rata-rata.

Outlier

outlier adalah nilai yang paling jauh dari rata-rata. Outlier juga bisa mengindikasikan bahwa suatu model akan mengalami **overfitting**, hal ini dikarenakan adanya outlier artinya menandakan **variasi** yang cukup banyak.



Outlier



Terdapat beberapa variabel yang memiliki **outlier** yaitu:

- targeted_productivity
- wip
- Incentive
- over_time

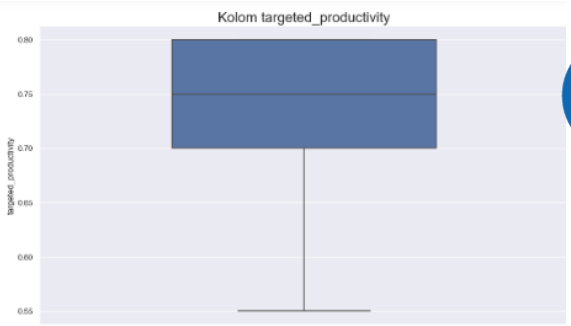
Handling Outlier

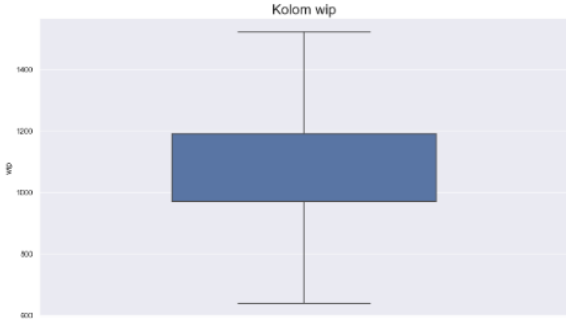
```python
def handle_outlier(df, variabel):
    Q1          = df[variabel].quantile(0.25)
    Q3          = df[variabel].quantile(0.75)
    IQR         = Q3 - Q1

    lower_limit = Q1 - 1.5 * IQR
    upper_limit = Q3 + 1.5 * IQR

    df.loc[df[variabel] > upper_limit, variabel] = upper_limit
    df.loc[df[variabel] < lower_limit, variabel] = lower_limit

    return df
```

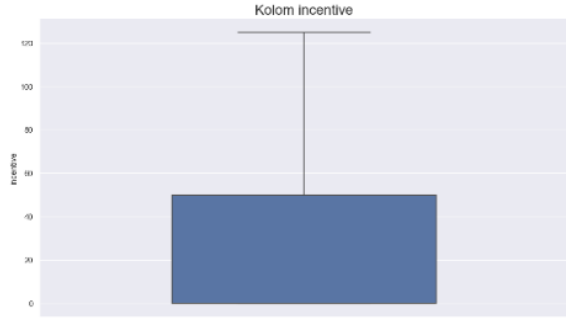
```python
variabel_outlir = ['targeted_productivity',
                   'incentive',
                   'wip',
                   'over_time']
```

```python
for var in variabel_outlir:
    df3  = handle_outlier(df3, var)
```

Mengganti nilai-nilai yang dianggap outlier atau nilai yang lebih dari lower limit dan kurang dari upper limit, supaya lebih dekat nilainya.







Exploratory Data Analysis

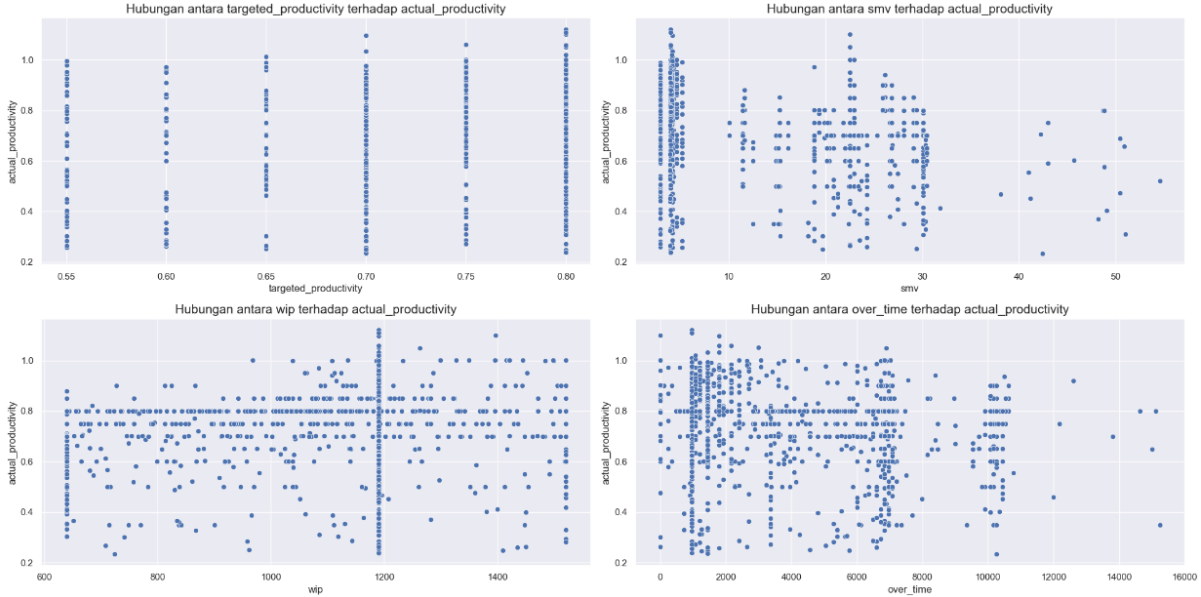
EDA on Numeric Data

Variabel target `actual_productivity` kita jadikan sebagai variabel **y** atau vertikal pada visualisasi dibawah ini, dikarenakan variabel tersebut merupakan **kontinyu**.

```python
numerik_vars2 = [
                 'targeted_productivity',
                 'smv',
                 'wip',
                 'over_time',
                 'incentive',
                 'idle_time',
                 'idle_men',
                 'no_of_style_change',
                 'no_of_workers']
```

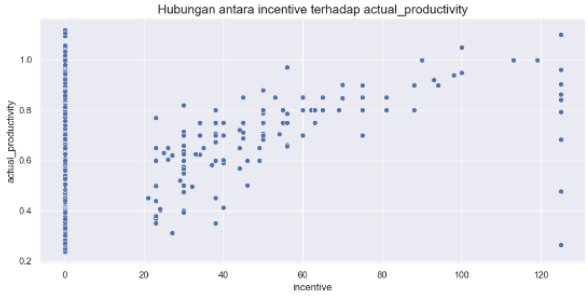
```python
def scatter_plot(df, numerik_vars, target_var):
    num               = df[numerik_vars]
    n                 = len(num.columns)
    rows              = n // 2 + n % 2
    cols              = 2
    sns.set(font_scale=1)
    fig, ax           = plt.subplots(rows, cols, figsize=(20, rows * 5))

    for i in range(rows):

        for j in range(cols):
            index     = i * cols + j

            if index  < n:

                col   = num.columns[index]
                sns.scatterplot(ax=ax[i, j],
                                data=df,
                                x=col,
                                y=target_var)

                ax[i, j].set_title(f'Hubungan antara {col} terhadap actual_productivity', fontdict={'fontsize': 15})
                ax[i, j].set_xlabel(col)
                ax[i, j].set_ylabel('actual_productivity')

    plt.tight_layout()
    plt.show()
```

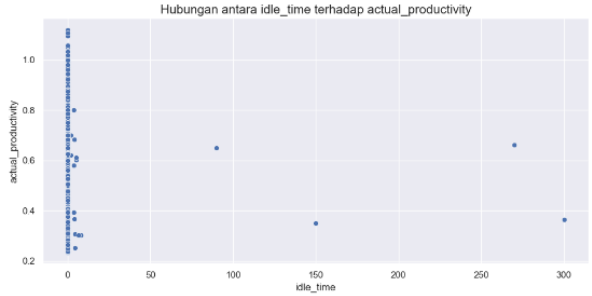
EDA on Numeric Data

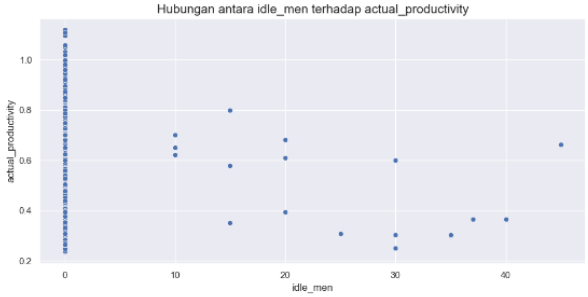


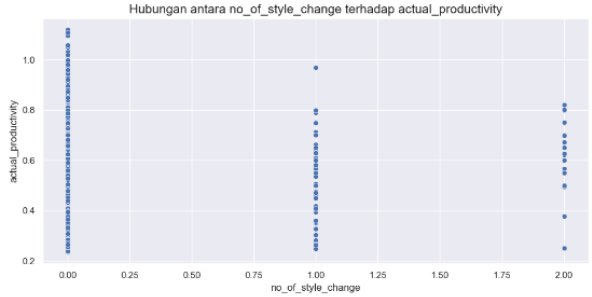
- Targeted_productivity: Target produktivitas 0.88 sering dicapai oleh produktivitas aktual.
- smv: SMV pada rentang 0-10 memiliki variasi produktivitas aktual yang tinggi, sedangkan SMV > 35 kurang produktif.
- wip: WIP yang lebih tinggi berkorelasi dengan sedikit peningkatan produktivitas aktual, tetapi tidak konsisten.
- over_time: Waktu lembur yang sedikit berkorelasi dengan produktivitas aktual yang lebih tinggi.

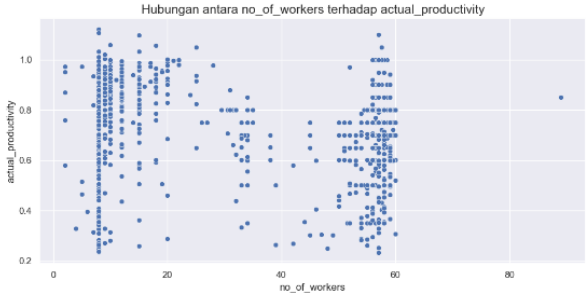
EDA on Numeric Data

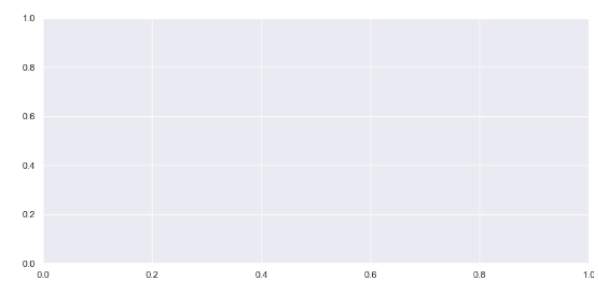






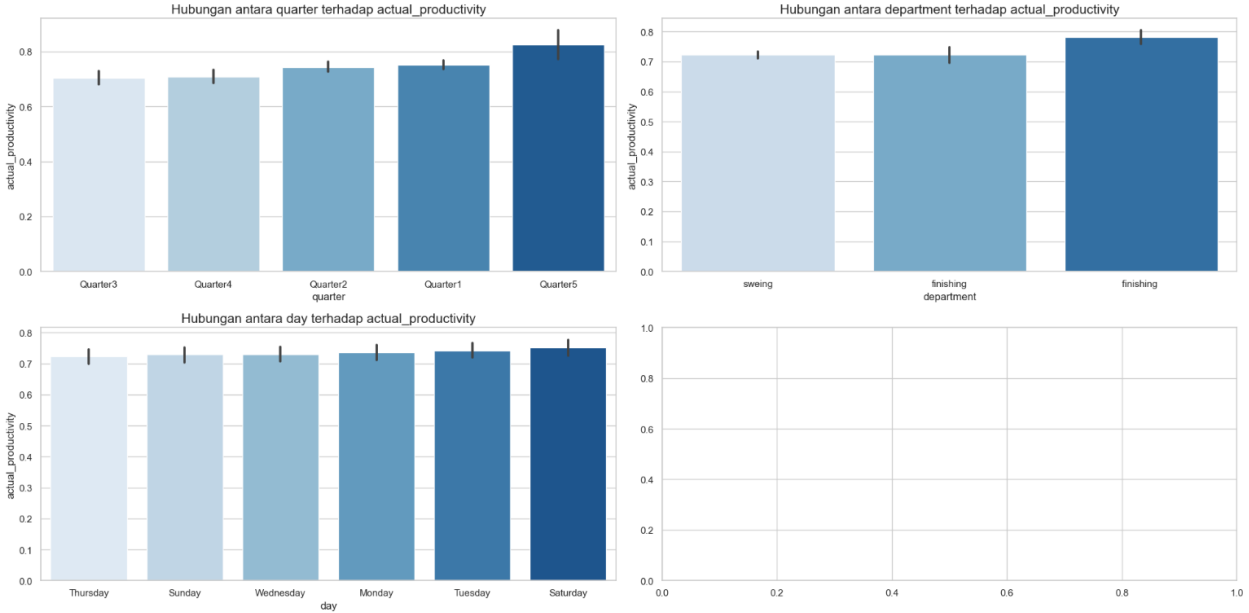






- Data incentive memiliki hubungan positif dengan actual_productivity
- Data idle_men memiliki hubungan negatif dengan actual_productivity, Artinya semakin banyak pekerja yang menganggur maka produktivitas aktual cenderung rendah.
- Data no_of_workers memiliki variasi yang beragam dan tidak menunjukkan pola yang jelas terhadap actual_productivity.
- Data idle_time tidak menunjukkan tren yang jelas terhadap actual_productivity karena persebaran datanya yang padat ketika bernilai 0.

EDA on Categoric Data



- Quarter 5 memiliki performa karyawan terbaik dengan nilai actual_productivity yang lebih tinggi dibandingkan dengan kuartal lainnya.
- Departemen finishing memiliki produktivitas aktual yang sedikit lebih tinggi dibandingkan dengan departemen sewing.
- Enam hari kerja dalam seminggu memiliki produktivitas aktual yang sama rata.

Modeling and Evaluation

Data Preprocessing

Data Preprocessing merupakan proses memperbaiki atau menghapus data yang salah, rusak, format yang salah, duplikat, atau tidak lengkap dalam kumpulan data. Intinya suatu tahapan dimana akan mempersiapkan variabel-variabel yang cocok untuk digunakan ketika **Modeling**. Tahapan-tahapan yang akan digunakan yaitu:

Feature Engineering:

- **Encoding**, merupakan teknik untuk mengubah data kategorikal menjadi data numerik.
- **Scaling**, merupakan teknik untuk mereduce nilai supaya tidak terlalu besar datanya. Supaya perbedaan datanya tidak terlalu besar antara suatu variabel.

Feature Selection:

- **Recursive Feature Elimination (RFE)**, metode iteratif yang memilih fitur-fitur terbaik dengan cara melakukan berulang kali membangun model dan menghapus fitur yang paling sedikit berkontribusi pada model.

Feature Engineering (FE)

Encoding

```
# 'Day'
one_hot_day                 = OneHotEncoder(sparse_output=False)
day_reshaped                = df3['day'].values.reshape(-1, 1)
day_one_hot                 = one_hot_day.fit_transform(day_reshaped)
day_encoded                 = pd.DataFrame(day_one_hot, columns=one_hot_day.get_feature_names_out(['day']))
df3_day                     = pd.concat([df3.drop(columns=['day']), day_encoded], axis=1)

# 'Quarter'
ordinal                     = OrdinalEncoder()
df3_day['quarter_encod']    = ordinal.fit_transform(df3_day[['quarter']])

# 'Department'
one_hot_department          = OneHotEncoder(sparse_output=False)
department_reshaped         = df3_day['department'].values.reshape(-1, 1)
department_one_hot          = one_hot_department.fit_transform(department_reshaped)
department_encoded          = pd.DataFrame(department_one_hot, columns=one_hot_department.get_feature_names_out(['department
df3_day                     = pd.concat([df3_day.drop(columns=['department']), department_encoded], axis=1)
```

Terdapat 3 variabel kategorik yang diterapkan *encoding* yaitu:

- `day` = Variabel kategorik nominal yang akan diterapkan one-hot encoding.
- `quarter` = Variabel katogorik ordinal yang diterapkan Ordinal Encoder.
- `department` = Variabel kategorik nominal yang akan diterapkan one-hot encoding.

Feature Engineering

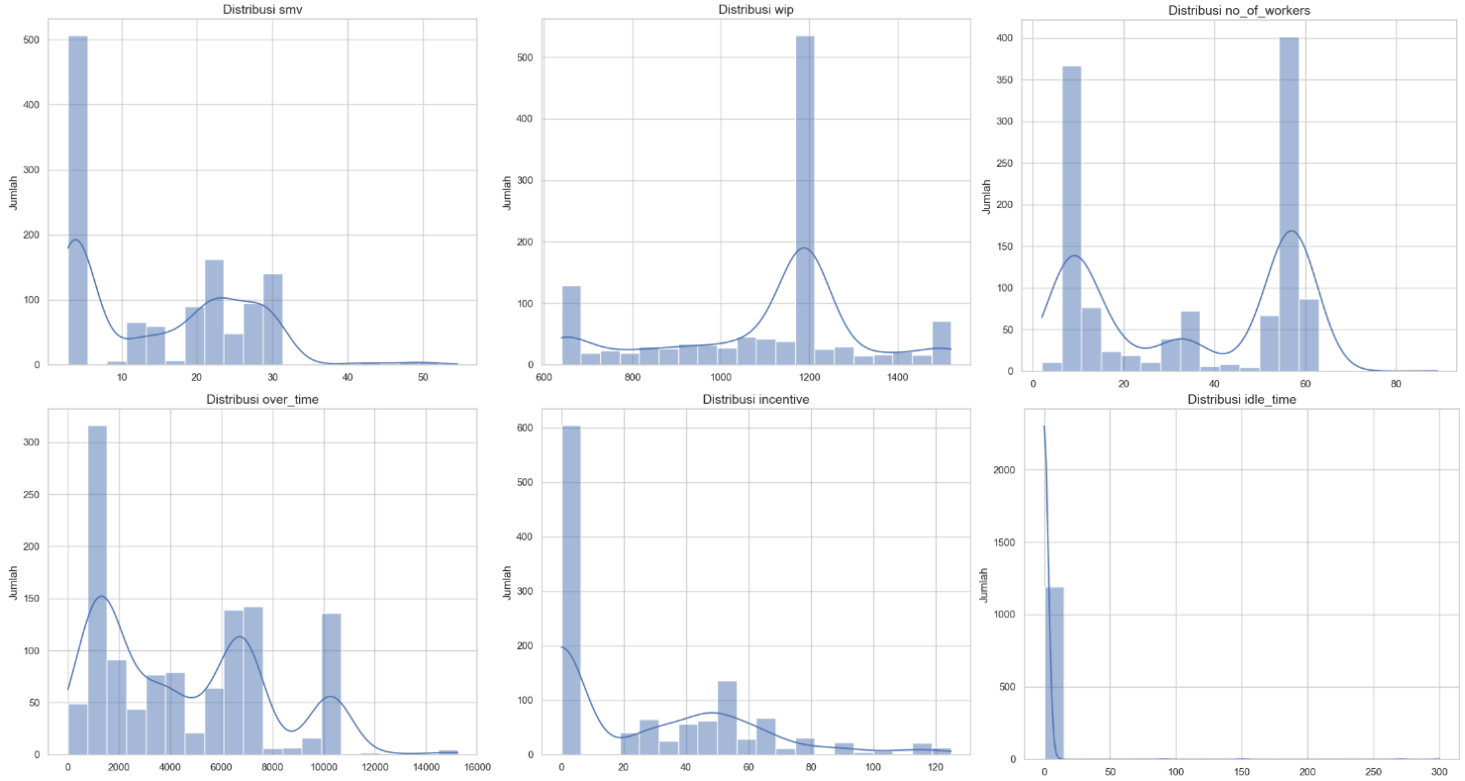
Result Encoding

```
1 df3_day
```

ay	day_Sunday	day_Thursday	day_Tuesday	day_Wednesday	quarter_encod	department_finishing	department_finishing	department_sweing
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...
0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0

FE: Scaling

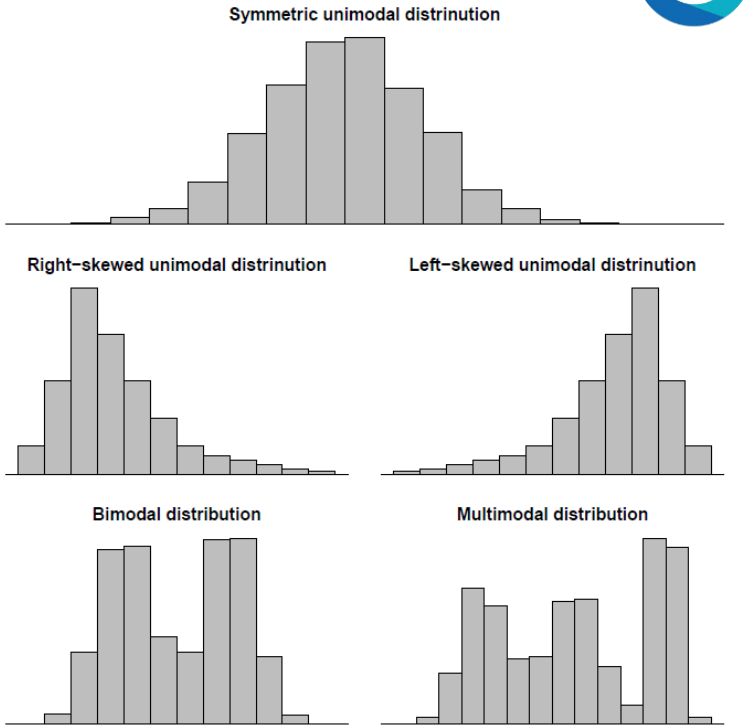
Memeriksa Distribusi Data



FE: Scaling

Teknik untuk mereduce nilai supaya tidak terlalu besar datanya dan berada dalam distribusi normal. Pentingnya data berdistribusi normal itu supaya mengurangi kemungkinan terjadinya overfitting dan underfitting.

- `smv` = distribusi sedikit skew
- `wip` = distribusi kurtosis
- `over_time` = distribusi normal
- `incentive` = distribusi skewness ke kanan
- `no_of_workers` = distribusi normal / Bimodial
- `idle_time` = distribusi skew



FE: Scaling

Implementasi Scaling

- `smv` = min-max scaling
- `wip` = robust scaling
- `overtime` = min-max scaling
- `incentive` = log transformation
- `no_of_workers` = min-max scaling
- `idle_time` = log transformation

```
smv_scaler              = MinMaxScaler()
wip_scaler              = RobustScaler()
over_time_scaler        = MinMaxScaler()
incentive_scaler        = np.log1p
no_of_workers_scaler    = MinMaxScaler()
```

```
X_skala                 = X.copy()
```

```
X_skala['smv']          = smv_scaler.fit_transform(X[['smv']])
```

```
X_skala['wip']          = wip_scaler.fit_transform(X[['wip']])
```

```
X_skala['over_time']    = over_time_scaler.fit_transform(X[['over_time']])
```

```
X_skala['incentive']    = incentive_scaler(X['incentive'])
```

```
X_skala['idle_time']    = incentive_scaler(X['idle_time'])
```

```
X_skala['no_of_workers'] = no_of_workers_scaler.fit_transform(X[['no_of_workers']])
```

Result FE

```
1 X
```

	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_change	no_of_workers	day_Monda
0	8	0.80	0.450252	-0.374053	0.464567	4.595120	0.0	0	0	0.655172	
1	1	0.75	0.020132	0.000000	0.062992	0.000000	0.0	0	0	0.068966	
2	11	0.80	0.164731	-1.009072	0.240157	3.931826	0.0	0	0	0.327586	
3	12	0.80	0.164731	-1.009072	0.240157	3.931826	0.0	0	0	0.327586	
4	6	0.80	0.445219	-0.092831	0.125984	3.931826	0.0	0	0	0.620690	
...	...	...	...	...	...	...	...	...	...	...	
1192	10	0.75	0.000000	0.000000	0.062992	0.000000	0.0	0	0	0.068966	
1193	8	0.70	0.019357	0.000000	0.062992	0.000000	0.0	0	0	0.068966	
1194	7	0.65	0.019357	0.000000	0.062992	0.000000	0.0	0	0	0.068966	
1195	9	0.75	0.000000	0.000000	0.118110	0.000000	0.0	0	0	0.149425	
1196	6	0.70	0.000000	0.000000	0.047244	0.000000	0.0	0	0	0.045977	

1197 rows × 20 columns

Feature Selection (FS)

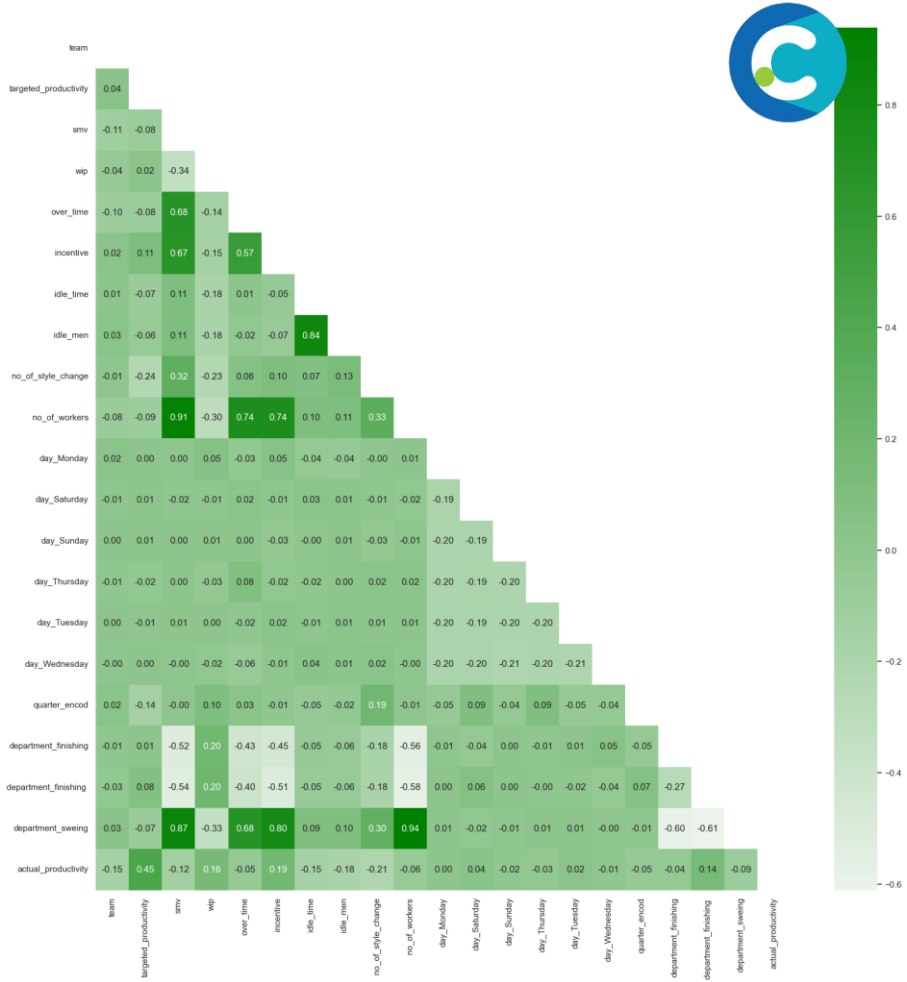
Visualisasi Korelasi

Variabel yang positif terhadap variabel target

- `targeted_productivity`
- `incentive`

Fitur-fitur tersebut masih belum cukup dan perlu dilakukan pencarian fitur terbaik, salah satu teknik yang digunakan yaitu **Recursive Feature Elimination (RFE)**.

Tujuan utamanya adalah untuk meningkatkan kinerja model dengan mengurangi jumlah fitur yang tidak relevan atau redundant.



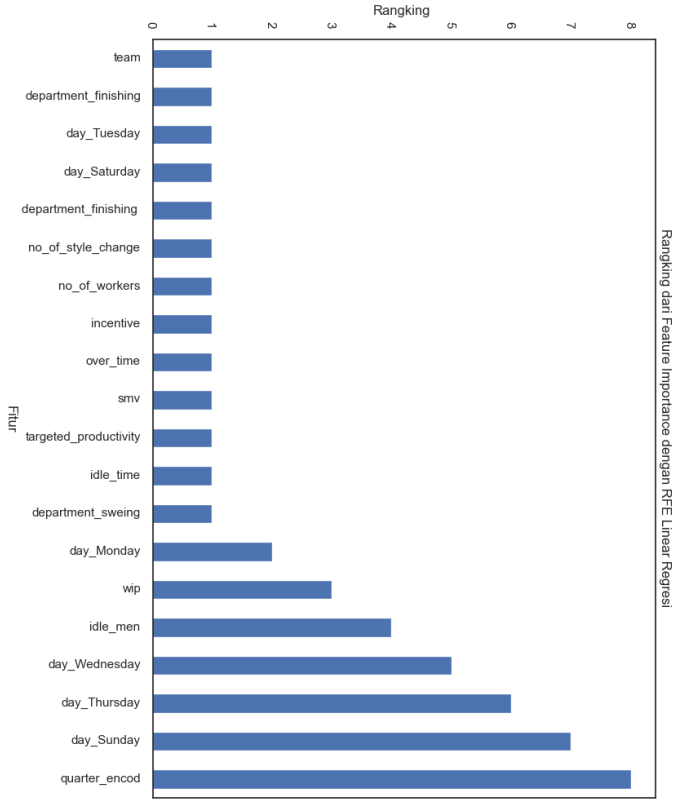
FS with RFE

Top 13 Fitur

- team
- department_finishing
- day_Tuesday
- day_Saturday
- department_finishing
- no_of_style_change
- no_of_workers
- incentive
- over_time
- smv
- targeted_productivity
- idle_time
- department_sweing

```
model_fitur_linear = LinearRegression()
n_fitur_pilih      = 13

rfe                = RFE(estimator=model_fitur_linear,
                         n_features_to_select=n_fitur_pilih)
```

```
rfe.fit(X, y)
```



Result Data Preprocessing

```
1 X_baru
```

	team	targeted_productivity	smv	over_time	incentive	idle_time	no_of_style_change	no_of_workers	day_Saturday	day_Tuesday
0	8	0.80	0.450252	0.464567	4.595120	0.0	0	0.655172	0.0	0.0
1	1	0.75	0.020132	0.062992	0.000000	0.0	0	0.068966	0.0	0.0
2	11	0.80	0.164731	0.240157	3.931826	0.0	0	0.327586	0.0	0.0
3	12	0.80	0.164731	0.240157	3.931826	0.0	0	0.327586	0.0	0.0
4	6	0.80	0.445219	0.125984	3.931826	0.0	0	0.620690	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
1192	10	0.75	0.000000	0.062992	0.000000	0.0	0	0.068966	0.0	0.0
1193	8	0.70	0.019357	0.062992	0.000000	0.0	0	0.068966	0.0	0.0
1194	7	0.65	0.019357	0.062992	0.000000	0.0	0	0.068966	0.0	0.0
1195	9	0.75	0.000000	0.118110	0.000000	0.0	0	0.149425	0.0	0.0
1196	6	0.70	0.000000	0.047244	0.000000	0.0	0	0.045977	0.0	0.0

1197 rows × 13 columns

Modeling

1. Split Data

```
X_train, X_test, y_train, y_test = train_test_split(X_baru,
                                                    y,
                                                    test_size=0.25,
                                                    random_state=42)
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(897, 13)
(300, 13)
(897, 1)
(300, 1)
```

Modeling

2. Initiate Model and Parameter

```
linear_regression       = LinearRegression()
knn_regressor           = KNeighborsRegressor(n_neighbors=30, algorithm="auto")  # 'auto' mencari algoritma terbaiknya
decision_tree_regressor = DecisionTreeRegressor(max_depth=10, criterion="squared_error")
random_forest_regressor = RandomForestRegressor(max_depth=15, criterion="squared_error", n_estimators=38)
svm_rbf_regressor       = SVR(kernel='rbf')
mlp_regressor           = MLPRegressor(hidden_layer_sizes=(100,), activation='relu', solver='adam', max_iter=500)
xgboost_regressor       = xgb.XGBRegressor()
lightgbm_regressor      = lgb.LGBMRegressor()
catboost_regressor      = CatBoostRegressor(verbose=0)
```

```
models = {
    "Linear Regression"       : linear_regression,
    "KNN Regression"          : knn_regressor,
    "Decision Tree Regression": decision_tree_regressor,
    "Random Forest Regression": random_forest_regressor,
    "SVM RBF Regression"      : svm_rbf_regressor,
    "MLP Regression"          : mlp_regressor,
    "XGBoost Regression"      : xgboost_regressor,
    "LightGBM Regression"     : lightgbm_regressor,
    "CatBoost Regression"     : catboost_regressor
}
```

Modeling & Evaluation

3. Fit or Train Model and Evaluate Model on Train Data

```python
print("*" * 46, "Performa Model ML pada Data Train", "*" * 46, "\n")

for nama_model, model in models.items():

    model.fit(X_train, y_train)
    y_pred_train = model.predict(X_train)

    print(f"{nama_model} Train: ")
    regression_model_report(y_train, y_pred_train)
    print("=" * 25 + "\n")
```

```
********************************************** Performa Model ML pada Data Train **********************************************

Linear Regression Train:

MAE: 0.1006
MSE: 0.0189
RMSE: 0.1375
sMAPE: 15.1281%
=========================

KNN Regression Train:

MAE: 0.1026
MSE: 0.0205
RMSE: 0.1433
sMAPE: 15.6127%
=========================

Decision Tree Regression Train:

MAE: 0.0345
MSE: 0.0056
RMSE: 0.0747
sMAPE: 5.3600%
=========================

Random Forest Regression Train:

MAE: 0.0341
MSE: 0.0036
RMSE: 0.0600
sMAPE: 5.6816%
=========================

SVM RBF Regression Train:

MAE: 0.1025
MSE: 0.0190
RMSE: 0.1379
sMAPE: 15.4921%
=========================

MLP Regression Train:

MAE: 0.1052
MSE: 0.0197
RMSE: 0.1404
sMAPE: 15.8469%
=========================

XGBoost Regression Train:

MAE: 0.0212
MSE: 0.0023
RMSE: 0.0475
sMAPE: 3.4092%
=========================

LightGBM Regression Train:

MAE: 0.0505
MSE: 0.0069
RMSE: 0.0831
sMAPE: 8.1434%
=========================

CatBoost Regression Train:

MAE: 0.0410
MSE: 0.0047
RMSE: 0.0689
sMAPE: 6.5490%
=========================
```

Modeling & Evaluation

4. Predict / Test Model and Evaluate Model on Test Data

```python
print("*"*50, "Performa Testing Model ML", "*"*50, "\n")

for nama_model, model in models.items():

    model.fit(X_train, y_train)
    y_pred_test = model.predict(X_test)

    print(f"{nama_model} Test: ")
    regression_model_report(y_test, y_pred_test)

    print("="*25 + "\n")
```

```
************************************************** Performa Testing Model ML **************************************************
Linear Regression Test:

MAE: 0.1015
MSE: 0.0210
RMSE: 0.1451
sMAPE: 15.3946%
=========================

Random Forest Regression Test:

MAE: 0.0778
MSE: 0.0170
RMSE: 0.1306
sMAPE: 12.1433%
=========================

XGBoost Regression Test:

MAE: 0.0846
MSE: 0.0198
RMSE: 0.1408
sMAPE: 12.8197%
=========================

KNN Regression Test:

MAE: 0.1062
MSE: 0.0220
RMSE: 0.1482
sMAPE: 15.8421%
=========================

SVM RBF Regression Test:

MAE: 0.1062
MSE: 0.0215
RMSE: 0.1467
sMAPE: 15.7859%
=========================

LightGBM Regression Test:

MAE: 0.0839
MSE: 0.0171
RMSE: 0.1308
sMAPE: 13.0323%
=========================

Decision Tree Regression Test:

MAE: 0.0963
MSE: 0.0271
RMSE: 0.1647
sMAPE: 15.1348%
=========================

MLP Regression Test:

MAE: 0.1045
MSE: 0.0228
RMSE: 0.1509
sMAPE: 15.9747%
=========================

CatBoost Regression Test:

MAE: 0.0814
MSE: 0.0172
RMSE: 0.1311
sMAPE: 12.5110%
=========================
```

Modeling & Evaluation

5. Result

- Linear Regression : Performa stabil (tidak overfitting/underfitting), namun tidak terlalu bagus.
- KNN Regression : Performa stabil, namun tidak terlalu bagus.
- Decision Tree Regression : Overfitting pada data test.
- **Random Forest Regression** : Cukup stabil pada data latih, sedikit menurun pada data test, perlu optimasi.
- SVM RBF Regression : Performa stabil (tidak overfitting/underfitting), namun tidak terlalu bagus.
- MLP Regression : Sedikit overfitting.
- **XGBoost Regression** : Good fit, namun perlu optimasi.
- LightGBM Regression : Good fit, namun perlu optimasi.
- **CatBoost Regression** : Performa terbaik (sedikit lebih baik dari LightGBM), namun perlu optimasi.

Recommendation

Recommendation Awal

Memilih Model Terbaik merupakan proses menentukan model atau algoritma manakah yang memiliki performa terbaik atau paling cocok terhadap memprediksi tingkat produktifitas aktual dari karyawan yang bekerja dalam perusahaan pakaian tersebut.

Setelah melihat performa awal dari ke-9 model sebelumnya berdasarkan nilai parameter MAE, MSE, RMSE, dan SMAPE. Maka dapat disimpulkan **Top 3 Model Awal** yang cocok adalah:

- Random Forest Regression
- XGBoost Regression
- CatBoost Regression

Ketiga model terbaik tersebut ditentukan berdasarkan:

- Nilai MAE, MSE, dan RMSE akan sangat bagus jika dibawah atau lebih kecil dari 0,1
- Nilai SMAPE yang dipilih itu pada rentant 10% - 15%

Hyperparameter Tuning

Top 3 Model Awal tersebut tentunya masih bisa dioptimalkan atau ditingkatkan performanya lagi dengan menggunakan teknik **Hyperparameter Tuning**. **Hyperparameter** adalah teknik untuk mencari parameter terbaik untuk mendapatkan model yang akurat.

Hyperparameter Tuning akan melakukan training dan prediksi dengan menggunakan data train dan validasi terhadap suatu model secara berulang kali dengan menggunakan kombinasi parameter yang berbeda-beda. Setelah itu, semua hasil percobaan training dan prediksi akan dipilih parameter terbaik yang digunakan dalam menghasilkan model dengan performa terbaik, hal ini disebut dengan **Cross Validation**.

Setiap model dibawah ini akan diterapkan teknik **RandomizedSearchCV**:

- Random Forest Regression
- CatBoost Regression
- XGBoost Regression

Hyperparameter Tuning

1. Random Forest dengan RandomizedSearchCV

```
rf_cv = RandomForestRegressor()

parameter_rf = {
    'n_estimators': randint(50, 200),
    'max_depth': [0, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt']
}

grid_cv_rf = RandomizedSearchCV(rf_cv,
                                param_distributions=parameter_rf,
                                n_iter=50,
                                scoring='neg_mean_squared_error',
                                cv=5,
                                verbose=0)
```

```
grid_cv_rf.best_params_
```

```
{'max_depth': 30,
 'max_features': 'sqrt',
 'min_samples_leaf': 1,
 'min_samples_split': 10,
 'n_estimators': 139}
```

```
rf_best = RandomForestRegressor(max_depth=20,
                                max_features='sqrt',
                                min_samples_leaf=1,
                                min_samples_split=10,
                                n_estimators=72)

rf_best.fit(X_train, y_train)
```

Hasil

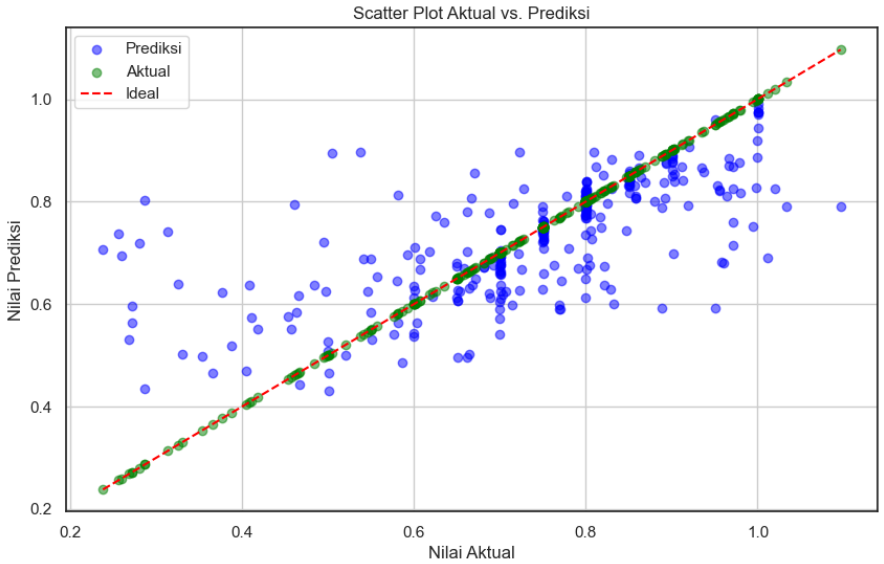
```
regression_model_report_viz(y_test, y_tt_rf_best)
```

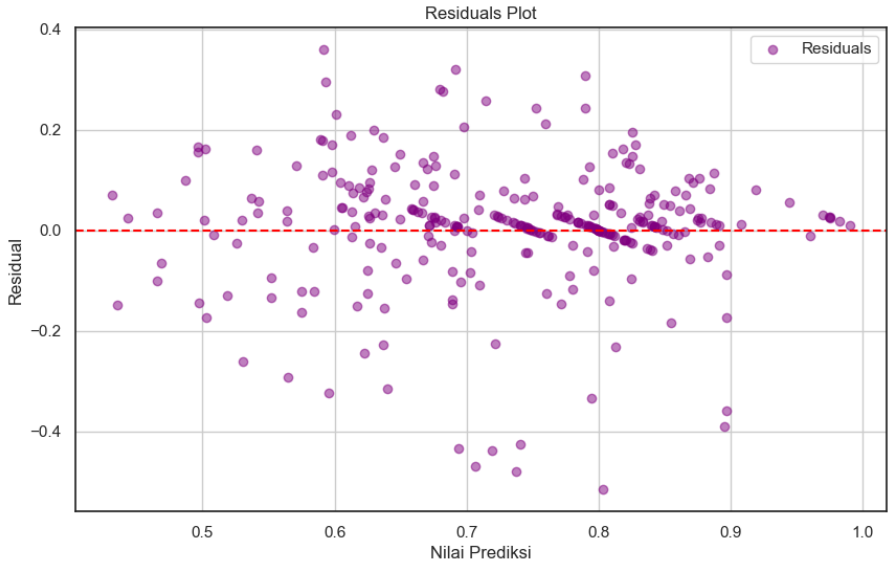
```
MAE: 0.0800
MSE: 0.0156
RMSE: 0.1250
sMAPE: 12.3539%
```

Hasil tersebut menunjukkan bahwa performa dari Random Forest semakin membaik setelah diterapkannnya Hyperparameter Tuning, yang dibuktikan oleh penurunan nilai MAE, MSE, dan RMSE.

Hyperparameter Tuning

1. Random Forest dengan RandomizedSearchCV





Hyperparameter Tuning

2. CatBoost Regressor dengan RandomizedSearchCV

```
cat_cv = CatBoostRegressor()

parameter_cat = {
    'learning_rate': np.linspace(0.01, 0.3),
    'depth': [4, 6, 8, 10],
    'l2_leaf_reg': [1, 3, 5, 7, 9],
    'iterations': randint(50, 200)
}

grid_cv_cat = RandomizedSearchCV(cat_cv,
                                 param_distributions=parameter_cat,
                                 n_iter=50,
                                 scoring='neg_mean_squared_error',
                                 cv=5,
                                 verbose=0)
```

```
grid_cv_cat.best_params_
```

```
{'depth': 6,
 'iterations': 179,
 'l2_leaf_reg': 5,
 'learning_rate': 0.14612244897959184}
```

```
cat_best = CatBoostRegressor(learning_rate=0.19
                             l2_leaf_reg=7,
                             depth=4,
                             iterations=165)

cat_best.fit(X_train, y_train)
```

Hasil

```
regression_model_report_viz(y_test, y_tt_cat_best)
```

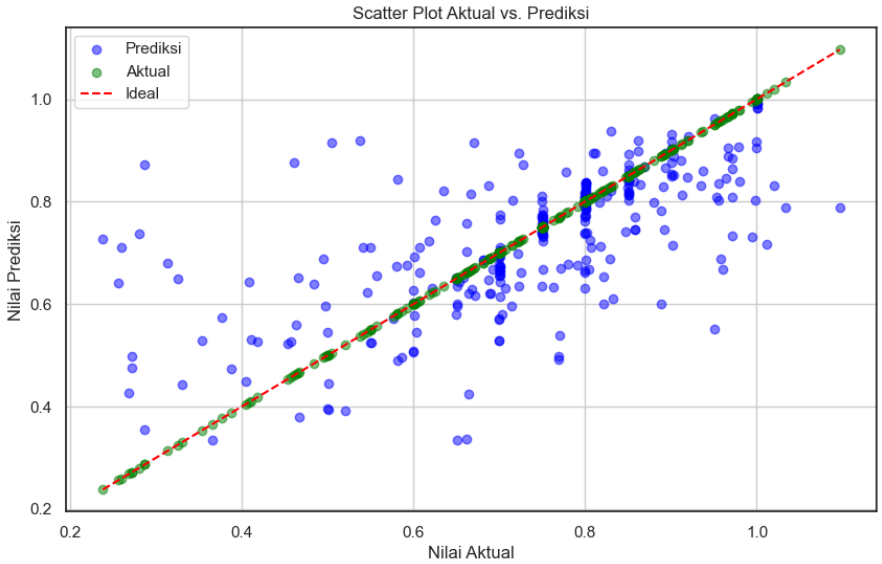
```
MAE: 0.0846
MSE: 0.0169
RMSE: 0.1300
sMAPE: 13.0961%
```

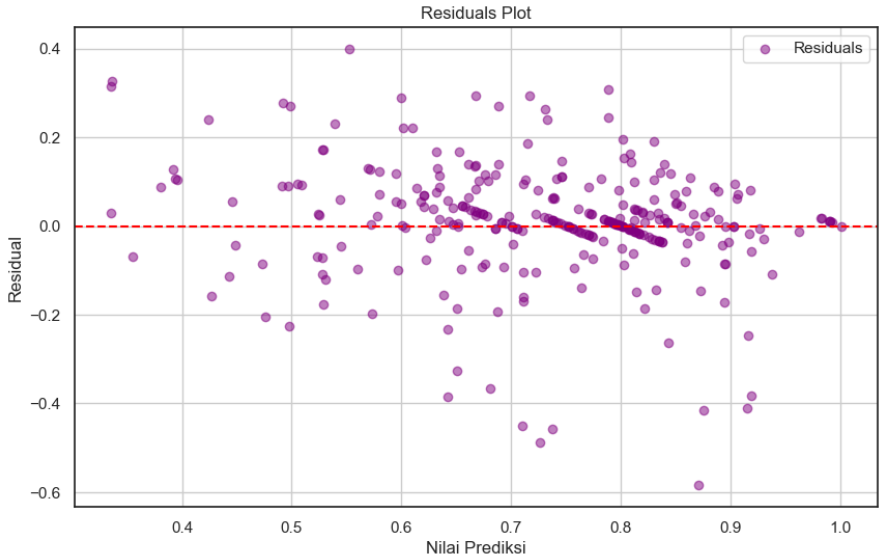
Hasil tersebut menunjukkan bahwa performa dari CatBoost semakin membaik setelah diterapkannnya Hyperparameter Tuning.

Namun performanya sedikit lebih tinggi jika dibandingkan dengan performa Random Forest. Sehingga membuat performa model CatBoost berada dibawah Random Forest.

Hyperparameter Tuning

2. CatBoost Regressor dengan RandomizedSearchCV





Hyperparameter Tuning

3. XGBoost Regressor dengan RandomizedSearchCV

```
xgb_cv = xgb.XGBRegressor()

parameter_xgb = {
    'learning_rate': np.linspace(0.01, 0.3, 30),
    'max_depth': [4, 6, 8, 10],
    'min_child_weight': [1, 3, 5, 7],
    'gamma': [0, 0.1, 0.2, 0.3],
    'colsample_bytree': np.linspace(0.3, 0.7, 5),
    'n_estimators': randint(50, 200)
}

grid_cv_xgb = RandomizedSearchCV(xgb_cv,
                                 param_distributions=parameter_xgb,
                                 n_iter=50,
                                 scoring='neg_mean_squared_error',
                                 cv=5,
                                 verbose=0)
```

```
grid_cv_xgb.best_params_
```

```
{'colsample_bytree': 0.7,
 'gamma': 0,
 'learning_rate': 0.09999999999999998,
 'max_depth': 4,
 'min_child_weight': 1,
 'n_estimators': 122}
```

```
xgb_best = xgb.XGBRegressor(colsample_bytree=0.4,
                            gamma=0,
                            learning_rate=0.11,
                            max_depth=4,
                            min_child_weight=3,
                            n_estimators=93)

xgb_best.fit(X_train, y_train)
```

Hasil

```
regression_model_report_viz(y_test, y_tt_xgb_best)
```

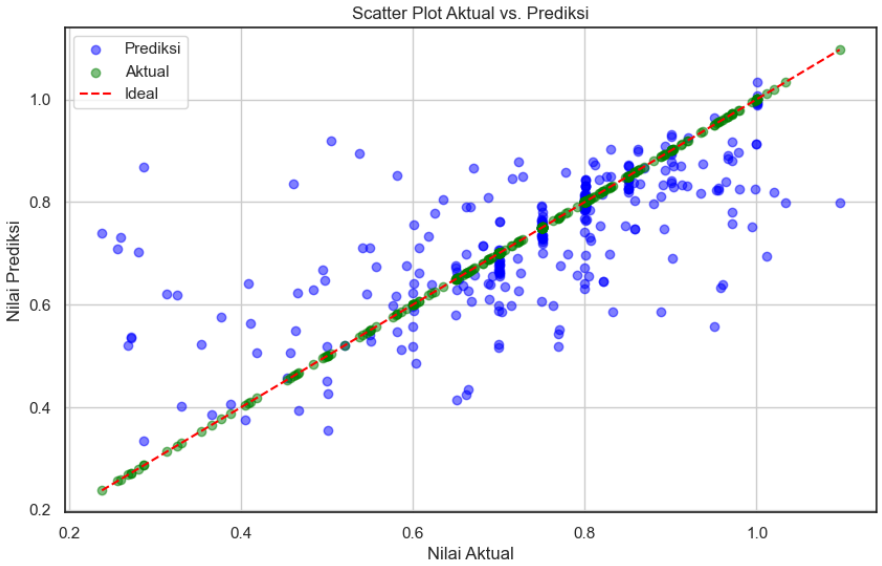
```
MAE: 0.0810
MSE: 0.0163
RMSE: 0.1276
sMAPE: 12.4588%
```

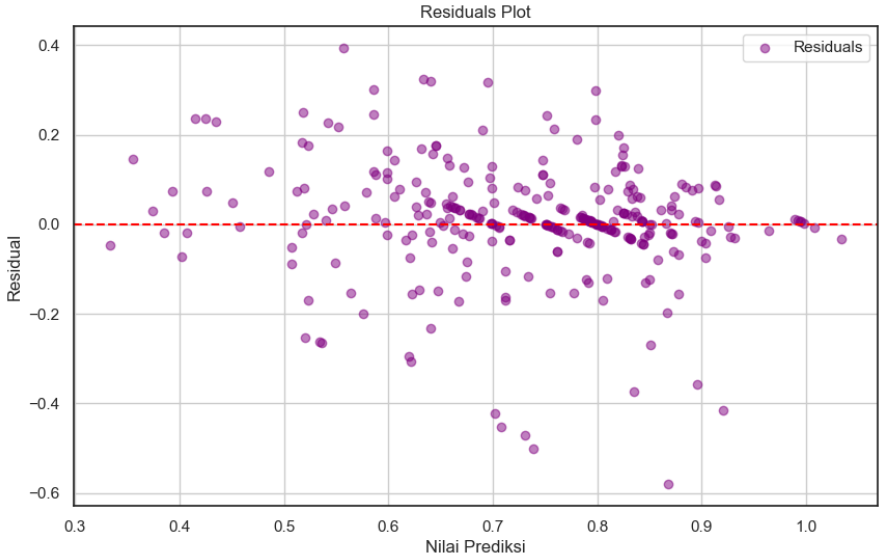
Hasil tersebut juga menunjukkan bahwa performa dari XGBoost yang semakin membaik.

Namun performanya sedikit lebih rendah jika dibandingkan dengan performa Cat Boost. Sehingga membuat performa model XGBoost lebih baik dibandingkan model CatBoost.

Hyperparameter Tuning

3. XGBoost Regressor dengan RandomizedSearchCV



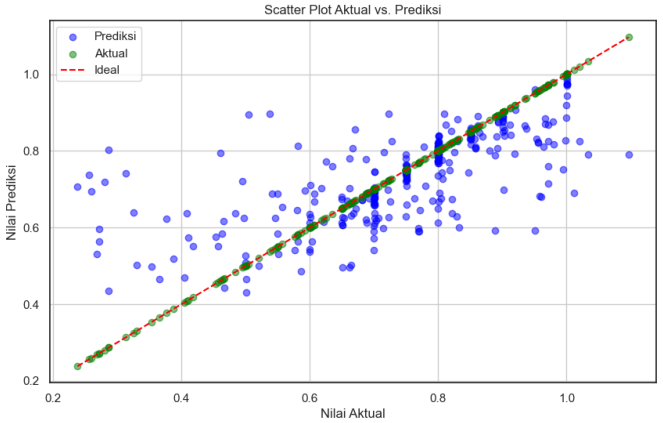


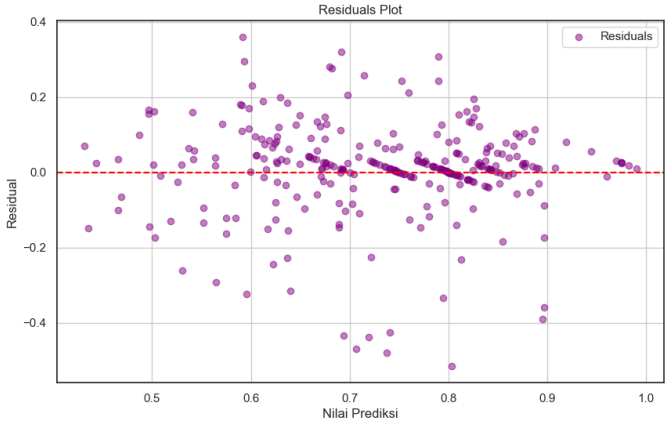
Recommendation

Model **Random Forest** setelah diterapkan Hyperparameter Tuning memiliki performa yang terbaik dalam memprediksi `actual_productivity` jika dibandingkan dengan model lainnya. Hal ini dibuktikan dengan nilai parameter evaluasi MAE, MSE, RMSE, dan SMAPE yang cukup rendah dan tidak menunjukkan adanya **overfitting** dan sedang dalam kondisi **good fit** sehingga menjadikan model Random Forest tersebut merupakan model terbaik.

```
1 regression_model_report_viz(y_test, y_tt_rf_best)
```

```
MAE: 0.0800
MSE: 0.0156
RMSE: 0.1250
sMAPE: 12.3539%
```





Challenges / Tantangan

Tantangan

Tantangan-tantangan yang dihadapi mulai dari tahapan cleaning, EDA, encoding, modelling antara lain:

- Terdapat missing values terhadap 50% jumlah record data, sehingga akan sangat disayangkan jika diisi dengan null atau dihapus.
- Banyak outlier pada beberapa kolom numerikal sehingga meningkatkan kemungkinan model overfitting.
- Variabel `team_no` merupakan nomor sebuah tim. Sehingga membutuhkan diskusi lebih lanjut terhadap pihak terkait apakah variabel `team_no` merupakan variabel kategorikal ordinal atau nominal?
- Beberapa model sepertinya mengalami overfitting, namun tidak saya gunakan dalam project ini. Namun bisa diatasi dengan hyperparameter tuning atau bisa kita improve dengan menerapkan teknik Regularization kedepannya.

Documentation

Berikut merupakan file-file yang diperlukan untuk mengakses informasi lebih lanjut:

- File Kode 1 : Preprocessing
- File Kode 2 : Modeling

Terima kasih

Thank you

谢谢

Xièxiè

ありがとうございまし

Arigatōgozaimashita

감사합니다

Kamsahamnida

cakap

PT. CERDAS DIGITAL NUSANTARA
Jl. Makaliwe Raya No. 36. Grogol,
Grogol Petamburan. Jakarta Barat 11450.