Untuk melihat codes, hasil, dan alasan dari langkah-langkah yang digunakan dapat dilihat secara lengkap melalui link colab berikut https://colab.research.google.com/drive/1kUfe36bRoWCmJmVnj8SaR6dp-5lEoRxD?usp=sharing

INPUT MODEL DAN DATA SET

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.decomposition import PCA
from statsmodels.stats.stattools import durbin_watson
from sklearn.preprocessing import QuantileTransformer
#!gdown 1yiQEpSMzYND8HBO_naAgM1e8q0FwPDmB #download data

!gdown 1Q3sigu3VvXSxkI2uxqWjGaeIMM4XWx6z #download dadta
!gdown 1EJzlaxjdvIiAe2nGXJBCly7Be5Y2RlCt #download peta

df = pd.read_excel('/content/datasatu.xlsx')
df.columns = df.columns.str.replace(' ', '_')
df['Provinsi'] = df['Provinsi'].map(lambda x: x.lower())
```

DATA INFORMATION

```
df.head()
df.info()
df.describe()
```

EDA

```
from mpl_toolkits.axes_grid1 import make_axes_locatable
from math import ceil
import geopandas as gpd

peta = gpd.read_file('./peta_id.geojson')
peta = peta.merge(df, left_on='WADMPR',
right_on='Provinsi').drop(columns='Provinsi')

feat = df.columns[2:]
y = df.columns[1]
```

```
ncol = 3

fig, ax = plt.subplots(1 + len(feat)//ncol + ceil(len(feat)%ncol/ncol),
ncol, figsize=(20,23), dpi=150)
gs = ax[0,0].get_gridspec()

for nowAx in ax[0, :]:
    nowAx.remove()

ax = np.delete(ax, 0, 0)
gs = fig.add_subplot(gs[0, :])

for nowAx in np.append(ax.flatten(), gs):
    nowAx.spines['top'].set_visible(False)
    nowAx.spines['bottom'].set_visible(False)
    nowAx.spines['left'].set_visible(False)
    nowAx.spines['right'].set_visible(False)

    nowAx.get_yaxis().set_visible(False)
    nowAx.get_xaxis().set_visible(False)

colors = ['PuBu', 'Reds', 'Greens', 'Oranges']

gs.set(title='P1')
divider = make_axes_locatable(gs)
cax = divider.append_axes("right", size="2.4%", pad=0.05)
peta.plot(column='P1',
          cmap=colors[-1],
         legend=True,
         ax=gs,
         cax=cax)

for i,j in enumerate(feat):
    nowAx = ax[i//ncol, i%ncol]
    nowAx.set(title=j)

    divider = make_axes_locatable(nowAx)
    cax = divider.append_axes("right", size="2.4%", pad=0.05)
    peta.plot(column=j,
              cmap=colors[i%len(colors)],
```

```
            legend=True,
            ax=nowAx,
            cax=cax)
sns.pairplot(df, y_vars=['P1'])
```

PREPROCESSING

Missing Values

```
missing_count = df.isnull().sum()
missing_percentage = (missing_count / len(df)) * 100

missing_percentage
```

Standarisasi

```
cols_to_process = ['Tingkat_Pengangguran', 'apm_sd', 'apm_smp',
                   'apm_sma', 'Berusaha_Sendiri',
                   'Berusaha_Dibantu_Buruh_Tidak_Tetap',
                   'Berusaha_Dibantu_Buruh_Tetap', 'Buruh',
                   'Pekerja_Bebas', 'Pekerja_Tidak_Dibayar',
                   'Upah_Per_Jam']
scaler = StandardScaler()
df[cols_to_process] = scaler.fit_transform(df[cols_to_process])
df.head()
```

Multikolinearitas

```
#Kolom provinsi tidak masuk karena kateogri dan P1 adalah variabel
dependen
X = df[[col for col in df.select_dtypes(include=['number']).columns
         if col not in ['P1', 'Provinsi' ]]]

# 2. Hitung VIF untuk setiap variabel
vif = pd.DataFrame()
vif["Variable"] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i)
                   for i in range(X.shape[1])]

# 3. Tampilkan hasil VIF
print(vif)
plt.figure(figsize=(10, 8)) # Mengatur ukuran heatmap
sns.heatmap(X.corr(), annot=True, cmap='coolwarm', fmt=".2f") # Membuat
heatmap
plt.title('Heatmap Korelasi Variabel X') # Menambahkan judul
plt.show() # Menampilkan heatmap
df = df.drop(['Pekerja_Tidak_Dibayar'], axis=1)
```

```
#Kolom kabupaten tidak masuk karena kateogri dan wpoor adalah variabel
dependen
X = df[[col for col in df.select_dtypes(include=['number']).columns
         if col not in ['P1', 'Provinsi']]]

# 2. Hitung VIF untuk setiap variabel
vif = pd.DataFrame()
vif["Variable"] = X.columns
vif["VIF"] = [variance_inflation_factor(X.values, i)
                   for i in range(X.shape[1])]

# 3. Tampilkan hasil VIF
print(vif)
```

SCATTER PLOT

```
y = df['P1']
X = df.drop(columns=['P1'])
plt.figure(figsize=(8, 6))
plt.scatter(X.iloc[:, 0], y, color='blue', alpha=0.7)
plt.title("Scatter Plot of X and Y")
plt.xlabel(X.columns[0])  # Set x-axis label to the first column name of X
plt.ylabel("Y")
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

OLS

Model 1

```
formula = 'P1 ~ Tingkat_Pengangguran + apm_sd + apm_smp + apm_sma +
Berusaha_Sendiri + Berusaha_Dibantu_Buruh_Tetap + Buruh + Pekerja_Bebas +
Upah_Per_Jam + Berusaha_Dibantu_Buruh_Tidak_Tetap'

# Membuat dan melatih model
model1 = smf.ols(formula=formula, data=df).fit()

# Menampilkan ringkasan model
print(model1.summary())
```

Transformasi Logaritma

```
from sklearn.preprocessing import QuantileTransformer
import numpy as np
import pandas as pd
```

```
# Inisialisasi transformer
transformer = QuantileTransformer(output_distribution='normal',
random_state=0, n_quantiles=38)

# Salin dataset
df_t = df.copy()

# Penanganan nilai inf, NaN, dan negatif
for column in ['Tingkat_Pengangguran', 'apm_sd', 'apm_smp', 'apm_sma',
               'Berusaha_Sendiri', 'Berusaha_Dibantu_Buruh_Tetap',
               'Berusaha_Dibantu_Buruh_Tidak_Tetap', 'Buruh',
               'Pekerja_Bebas', 'Upah_Per_Jam']:
    df_t[column] = df_t[column].replace([np.inf, -np.inf], np.nan)  #
Ganti inf dengan NaN
    df_t[column].fillna(0, inplace=True)  # Ganti NaN dengan 0
    df_t[column] = np.maximum(df_t[column], 0)  # Pastikan nilai negatif
menjadi 0

# Transformasi log
df_t['ln_P1'] = np.log(df_t['P1'] + 1)
df_t['ln_Tingkat_Pengangguran'] = np.log(df_t['Tingkat_Pengangguran'] + 1)
df_t['ln_apm_sd'] = np.log(df_t['apm_sd'] + 1)
df_t['ln_apm_smp'] = np.log(df_t['apm_smp'] + 1)
df_t['ln_apm_sma'] = np.log(df_t['apm_sma'] + 1)
df_t['ln_Berusaha_Sendiri'] = np.log(df_t['Berusaha_Sendiri'] + 1)
df_t['ln_Berusaha_Dibantu_Buruh_Tetap'] =
np.log(df_t['Berusaha_Dibantu_Buruh_Tetap'] + 1)
df_t['ln_Berusaha_Dibantu_Buruh_Tidak_Tetap'] =
np.log(df_t['Berusaha_Dibantu_Buruh_Tidak_Tetap'] + 1)
df_t['ln_Buruh'] = np.log(df_t['Buruh'] + 1)
df_t['ln_Pekerja_Bebas'] = np.log(df_t['Pekerja_Bebas'] + 1)
df_t['ln_Upah_Per_Jam'] = np.log(df_t['Upah_Per_Jam'] + 1)

# Kolom log-transform yang akan ditransformasi lebih lanjut
log_columns = [
    'ln_Tingkat_Pengangguran', 'ln_apm_sd', 'ln_apm_smp', 'ln_apm_sma',
    'ln_Berusaha_Sendiri', 'ln_Berusaha_Dibantu_Buruh_Tetap', 'ln_Buruh',
    'ln_Pekerja_Bebas', 'ln_Upah_Per_Jam',
'ln_Berusaha_Dibantu_Buruh_Tidak_Tetap'
```

```
]

# Terapkan QuantileTransformer
df_t[log_columns] = transformer.fit_transform(df_t[log_columns])
```

Model 2

```
# Formula dengan nama kolom yang tepat menggunakan tanda kutip
formula = 'P1 ~ ln_Tingkat_Pengangguran + ln_apm_sd + ln_apm_smp +
ln_apm_sma + ln_Berusaha_Sendiri + ln_Berusaha_Dibantu_Buruh_Tetap +
ln_Buruh + ln_Pekerja_Bebas + ln_Upah_Per_Jam +
ln_Berusaha_Dibantu_Buruh_Tidak_Tetap'

# Membuat dan melatih model
model2 = smf.ols(formula=formula, data=df_t).fit()

# Menampilkan ringkasan model
print(model2.summary())
```

Model 3

```
#LN P1
formula = 'ln_P1 ~ Tingkat_Pengangguran + apm_sd + apm_smp + apm_sma +
Berusaha_Sendiri + Berusaha_Dibantu_Buruh_Tetap + Buruh + Pekerja_Bebas +
Upah_Per_Jam + Berusaha_Dibantu_Buruh_Tidak_Tetap'

# Membuat dan melatih model
model3 = smf.ols(formula=formula, data=df_t).fit()

# Menampilkan ringkasan model
print(model3.summary())
```

Model 4

```
#SEMUA LN WEH
# Formula dengan nama kolom yang tepat menggunakan tanda kutip
formula = 'ln_P1 ~ ln_Tingkat_Pengangguran + ln_apm_sd + ln_apm_smp +
ln_apm_sma + ln_Berusaha_Sendiri + ln_Berusaha_Dibantu_Buruh_Tetap +
ln_Buruh + ln_Pekerja_Bebas + ln_Upah_Per_Jam +
ln_Berusaha_Dibantu_Buruh_Tidak_Tetap'

# Membuat dan melatih model
model4 = smf.ols(formula=formula, data=df_t).fit()
```

```
# Menampilkan ringkasan model
print(model4.summary())
```

Skewness

```
y = df['P1']
X = df.drop(columns=['P1'])
from scipy.stats import skew
numeric_cols = X.select_dtypes(include=np.number).columns
skew_x = X[numeric_cols].apply(skew)
df['P1'] = pd.to_numeric(df['P1'], errors='coerce')
skew_y = skew(df['P1'], nan_policy='omit')

print(f"Skewness X:\n{skew_x}")
print(f"Skewness Y:\n {skew_y}")
```

Best Model

```
model = {
    'Model': ['model1', 'model2', 'model3', 'model4'],
    'R-squared': [model1.rsquared, model2.rsquared, model3.rsquared,
                  model4.rsquared],
    'Adj. R-squared': [model1.rsquared_adj, model2.rsquared_adj,
model3.rsquared_adj,
                       model4.rsquared_adj],
    'AIC': [model1.aic, model2.aic, model3.aic,
            model4.aic],
    'BIC': [model1.bic, model2.bic, model3.bic,
            model4.bic],
    'F-statistic': [model1.fvalue, model2.fvalue, model3.fvalue,
                    model4.fvalue]
}

model_comparison = pd.DataFrame(model)

print(model_comparison)
```