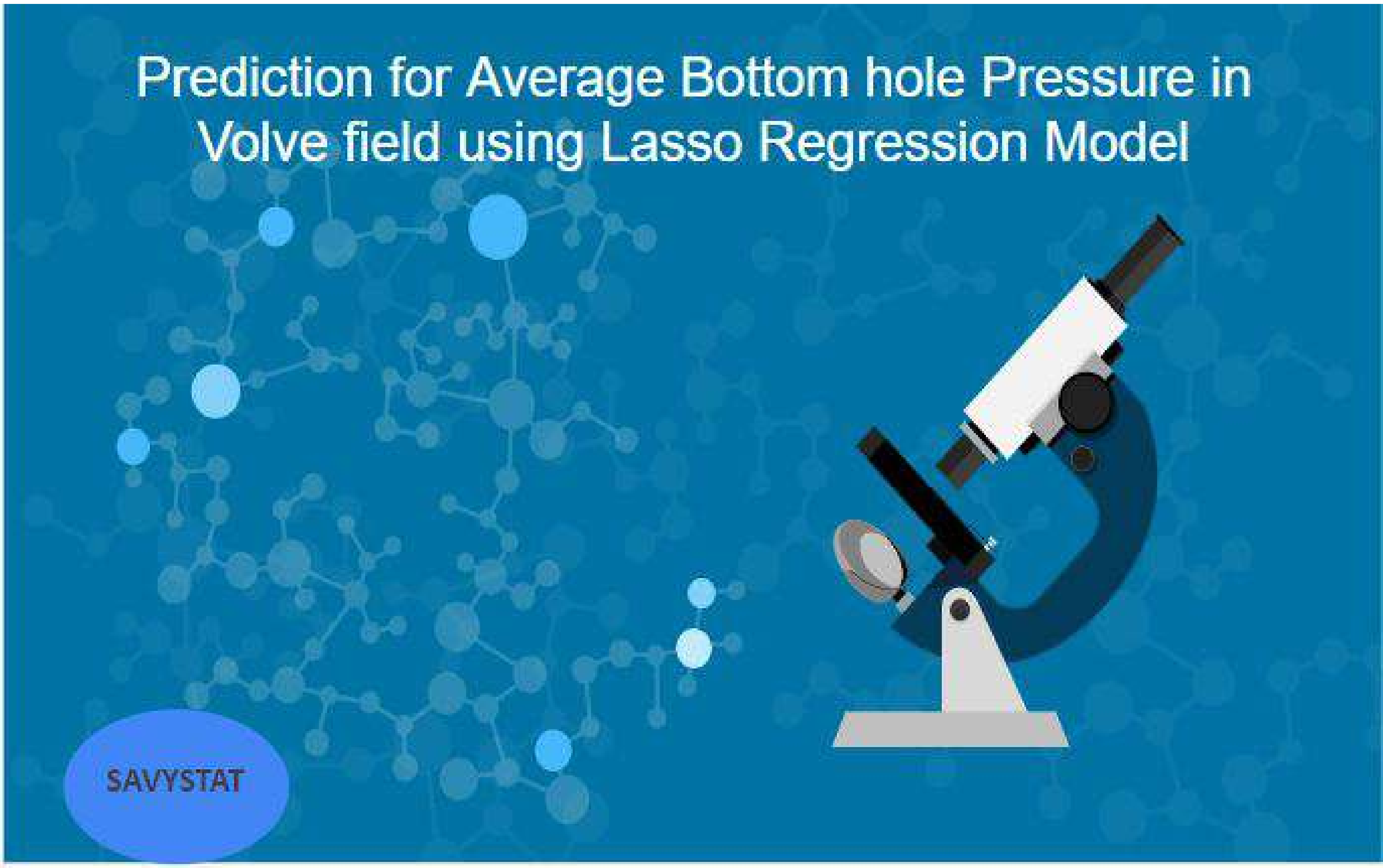


Prediction for Average Bottom hole Pressure in Volve field using Lasso Regression Model

SAVYSTAT

```
import pandas as pd
import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

```
df = pd.read_excel("Volve field dataset.xlsx")
```

```
df.head()
for column_name in df.columns:
    print(column_name)
```

```
ON_STREAM_HRS
AVG_DOWNHOLE_TEMPERATURE
AVG_ANNULUS_PRESS
AVG_CHOKE_SIZE_P
AVG_WHP_P
AVG_WHT_P
DP_CHOKE_SIZE
BORE_OIL_VOL
BORE_GAS_VOL
AVG_DOWNHOLE_PRESSURE
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15634 entries, 0 to 15633
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   ON_STREAM_HRS             15349 non-null  float64
 1   AVG_DOWNHOLE_TEMPERATURE  8980 non-null   float64
 2   AVG_ANNULUS_PRESS         7890 non-null   float64
 3   AVG_CHOKE_SIZE_P          8919 non-null   float64
 4   AVG_WHP_P                 9155 non-null   float64
 5   AVG_WHT_P                 9146 non-null   float64
 6   DP_CHOKE_SIZE             15340 non-null  float64
 7   BORE_OIL_VOL              9161 non-null   float64
 8   BORE_GAS_VOL              9161 non-null   float64
 9   AVG_DOWNHOLE_PRESSURE     8980 non-null   float64
dtypes: float64(10)
memory usage: 1.2 MB
```

```
# Check for missing values
missing_values = df.isnull().sum()
print(missing_values)
```

```
ON_STREAM_HRS                285
AVG_DOWNHOLE_TEMPERATURE    6654
AVG_ANNULUS_PRESS           7744
AVG_CHOKE_SIZE_P            6715
AVG_WHP_P                   6479
AVG_WHT_P                   6488
DP_CHOKE_SIZE                294
BORE_OIL_VOL                6473
BORE_GAS_VOL                6473
AVG_DOWNHOLE_PRESSURE       6654
dtype: int64
```

```
# Drop rows containing zeros
df = df[(df != 0).all(axis=1)]
# # Remove rows with any missing values
df_cleaned = df.dropna()
df_cleaned
```

	ON_STREAM_HRS	AVG_DOWNHOLE_TEMPERATURE	AVG_ANNULUS_PRESS	AVG_CHOKE_SIZE_P
762	7.00000	0.352200	2.885360	3.256548
763	24.00000	60.315740	19.464510	8.549131
769	5.07514	105.551370	21.550252	2.540804
772	15.07486	104.933215	1.652926	6.116182
773	24.00000	105.439765	17.308850	9.951288
...	...	...	...	...
8923	24.00000	106.517574	21.318431	31.575767
8924	24.00000	106.515586	21.105330	31.540612
8925	24.00000	106.521356	21.353661	31.522096
8926	24.00000	106.506781	20.629658	31.523457
8927	18.37500	106.507232	20.404848	24.922565

4166 rows × 10 columns

Next steps: Generate code with df_cleaned View recommended plots

```
missing_values = df_cleaned.isnull().sum()
print(missing_values)
```

```
ON_STREAM_HRS               0
AVG_DOWNHOLE_TEMPERATURE    0
AVG_ANNULUS_PRESS           0
AVG_CHOKE_SIZE_P            0
AVG_WHP_P                   0
AVG_WHT_P                   0
DP_CHOKE_SIZE               0
BORE_OIL_VOL                0
BORE_GAS_VOL                0
AVG_DOWNHOLE_PRESSURE       0
dtype: int64
```

```
df = df_cleaned
```

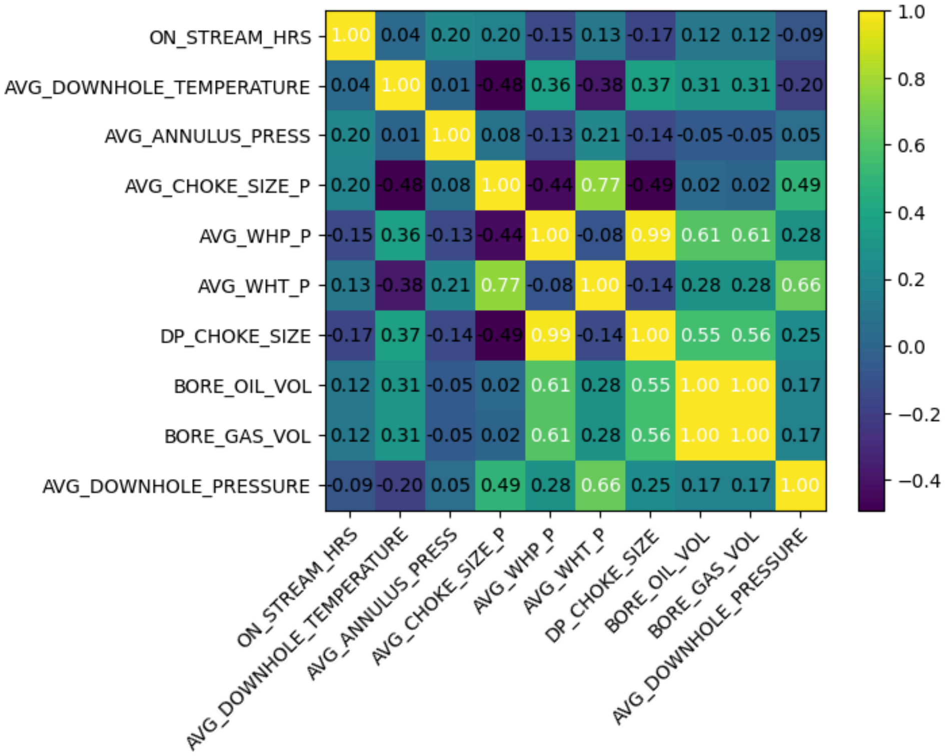
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4166 entries, 762 to 8927
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   ON_STREAM_HRS             4166 non-null   float64
 1   AVG_DOWNHOLE_TEMPERATURE  4166 non-null   float64
 2   AVG_ANNULUS_PRESS         4166 non-null   float64
 3   AVG_CHOKE_SIZE_P          4166 non-null   float64
 4   AVG_WHP_P                 4166 non-null   float64
 5   AVG_WHT_P                 4166 non-null   float64
 6   DP_CHOKE_SIZE             4166 non-null   float64
 7   BORE_OIL_VOL              4166 non-null   float64
 8   BORE_GAS_VOL              4166 non-null   float64
 9   AVG_DOWNHOLE_PRESSURE     4166 non-null   float64
dtypes: float64(10)
memory usage: 358.0 KB
```

```
df.describe()
```

	ON_STREAM_HRS	AVG_DOWNHOLE_TEMPERATURE	AVG_ANNULUS_PRESS	AVG_CHOKE_SIZE_P
count	4166.000000	4166.000000	4166.000000	4166.000000
mean	23.176426	104.105784	18.317129	53.508996
std	3.227762	4.097660	5.121428	37.138717
min	0.250000	0.352200	0.000020	0.600000
25%	24.000000	100.475027	14.382808	12.310157
50%	24.000000	105.944562	18.879005	50.840645
75%	24.000000	106.481651	22.065755	100.000000
max	25.000000	107.507552	30.019828	100.000000

```
import matplotlib.pyplot as plt
from mlxtend.plotting import heatmap
cols = df.columns.tolist()
cm = np.corrcoef(df[cols].values.T)
hm = heatmap(cm, row_names = cols, column_names = cols)
# Enlarge the figure size
plt.figure(figsize=(25, 20))
plt.show()
```



```
<Figure size 2500x2000 with 0 Axes>
```

```
x = df.iloc[:,:-1]
y = df.iloc[:, -1]
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y, random_state=0)
```

```
x_train.shape
```

```
(3124, 9)
```

```
y_train.shape
```

```
(3124,)
```

```
import numpy as np

# Take the natural logarithm of the features
x_train_log = np.log(x_train)
x_test_log = np.log(x_test)
```

```
import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso

# Generate a range of alpha values spanning multiple orders of magnitude
alphas = np.logspace(-3, 3, 7)  # This will generate values from 0.001 to 1000

# Create a pipeline with feature scaling and Lasso regression
pipeline = Pipeline([
    ('scaler', StandardScaler()),  # Feature scaling
    ('lasso', Lasso())  # Lasso regression
])

# Define hyperparameters to tune
param_grid = {'lasso__alpha': alphas}

# Perform grid search with 5-fold cross-validation
grid_search = GridSearchCV(estimator=pipeline, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error')

# Fit grid search to the training data
grid_search.fit(x_train_log, y_train)  # Replace x_train_scaled and y_train with your training data

# Get the best alpha value
best_alpha = grid_search.best_params_['lasso__alpha']

# Print the best alpha value
print("Best alpha:", best_alpha)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective
  model = cd_fast.enet_coordinate_descent(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_coordinate_descent.py:631: ConvergenceWarning: Objective
  model = cd_fast.enet_coordinate_descent(
Best alpha: 0.1
```

```
#fit the model
from sklearn.linear_model import Lasso
lasso_mod = Lasso(alpha=0.1).fit(x_train_log,y_train)
```

```
# lasso_mod = Lasso(alpha=0.01, max_iter=10000).fit(x_train_log, y_train)
```

```
lasso_mod.coef_
```

```
array([  2.03044613,  30.39507044,  -0.33456594,  16.662728  ,
        50.75392338,  38.25917017,   3.93565611, -18.32982729,
        -0.        ])
```

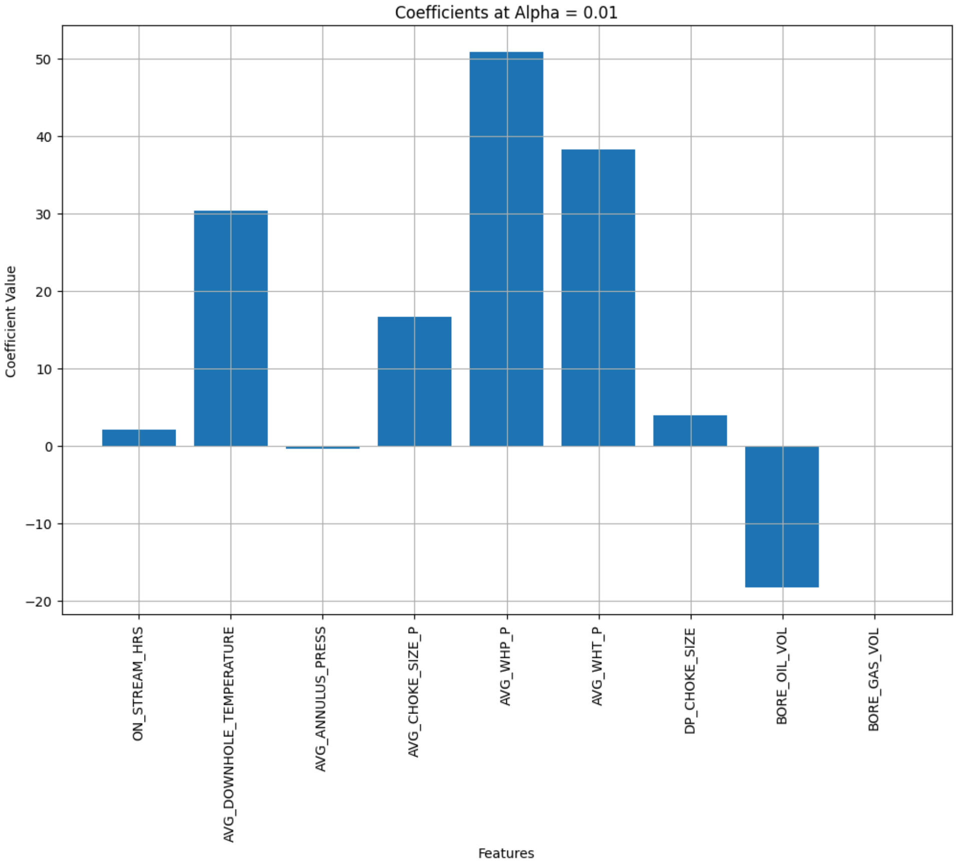
```
print("Intercept:", lasso_mod.intercept_)
```

```
Intercept: -206.58070093289825
```

```
# identifying which of the features contribute to the model
# Get feature names
feature_names = df.columns
coefficients = lasso_mod.coef_

# Create DataFrame to store coefficients and feature names
coefficients_df = pd.DataFrame({'Feature': feature_names[:-1], 'Coefficient': coefficients})
```

```
# Plot coefficients
plt.figure(figsize=(12, 8))
plt.bar(coefficients_df['Feature'], coefficients_df['Coefficient'])
plt.xlabel('Features')
plt.ylabel('Coefficient Value')
plt.title('Coefficients at Alpha = 0.01')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.grid(True)
plt.show()
```



```
#printing the r-squared on training data
print('R- squared on training data', lasso_mod.score(x_train_log, y_train)*100)
```

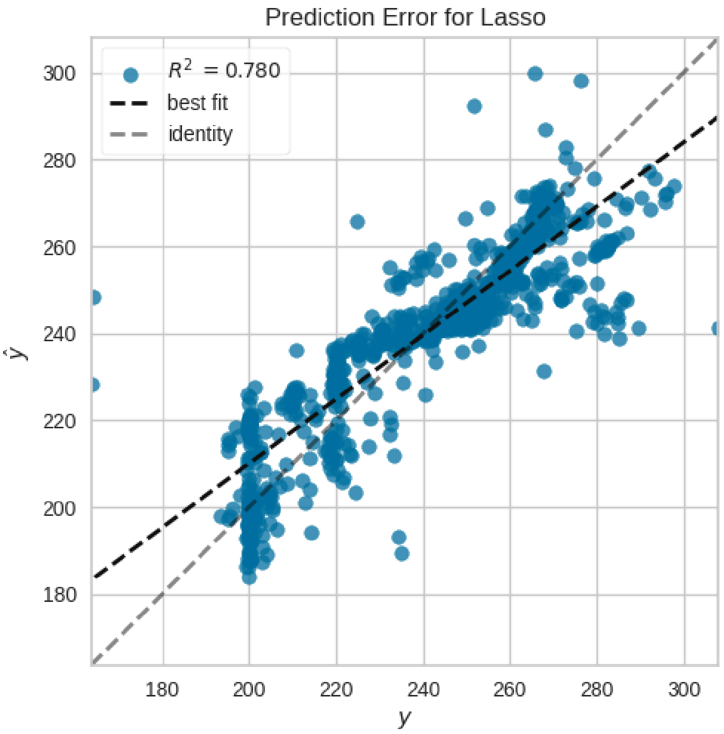
```
     R- squared on training data 76.58081637841275
```

```
#printing the r-squared on test data
print('R- squared on test data', lasso_mod.score(x_test_log, y_test)*100)
```

```
     R- squared on test data 78.00258393658079
```

```
#plotting the prediction errors and residuals using yellow brick
!pip install yellowbrick
from sklearn.preprocessing import StandardScaler
from yellowbrick.regressor import PredictionError, ResidualsPlot
visualizer = PredictionError(lasso_mod)
visualizer.fit(x_train_log, y_train)
visualizer.score(x_test_log,y_test)
visualizer.poof()
```

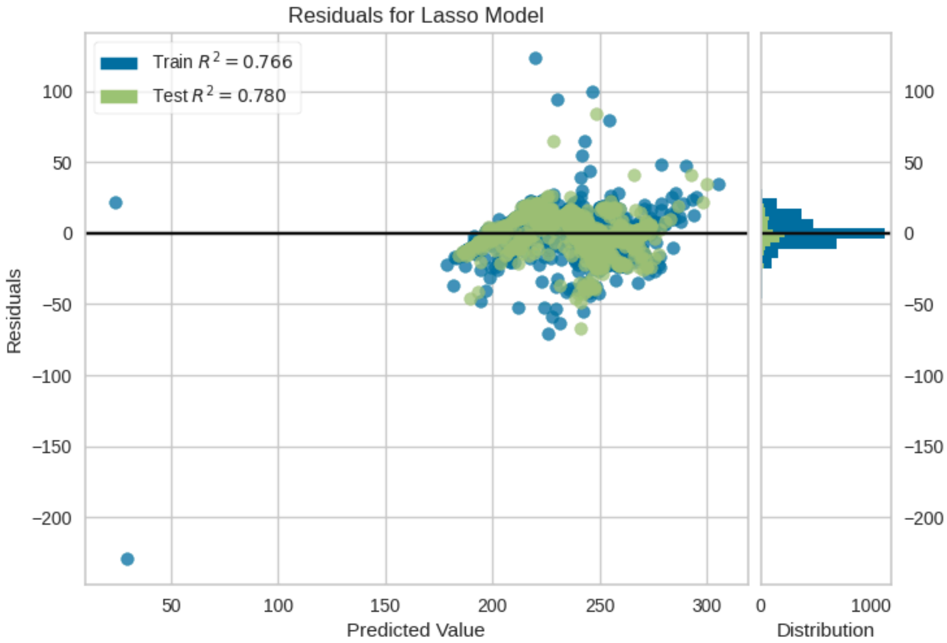
```
Requirement already satisfied: yellowbrick in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in /usr/local/lib/python3.10
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: cycler>=0.10.0 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (f
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not
  warnings.warn(
```



```
<Axes: title={'center': 'Prediction Error for Lasso'}, xlabel='$y$',
ylabel='$\\hat{y}$'>
```

```
#plotting the residuals
visualizer = ResidualsPlot(lasso_mod)
visualizer.fit(x_train_log,y_train)
visualizer.score(x_test_log,y_test)
visualizer.poof()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not
  warnings.warn(
```



```
<Axes: title={'center': 'Residuals for Lasso Model'}, xlabel='Predicted Value',
ylabel='Residuals'>
```

```
y_pred =lasso_mod.predict(x_test_log)


# Create a DataFrame to display the predicted values for the test data
result_df = pd.DataFrame({'Predicted Values': y_pred})


# Create a DataFrame to display the predicted and actual values
result_df = pd.DataFrame({'Actual Values': y_test, 'Predicted Values of AVG_DOWNHOLE_PRESSURE': y_pred})

# Display the DataFrame
print(result_df.head(15))
```

```
      Actual Values  Predicted Values of AVG_DOWNHOLE_PRESSURE
1163     265.812269                                 245.906340
8087     199.714866                                 189.428519
7184     265.382097                                 262.564390
2170     235.708089                                 252.861815
8683     223.130863                                 212.361288
2859     267.022065                                 253.122667
1860     251.343919                                 241.469494
7640     267.210376                                 267.793313
7527     266.157293                                 261.145763
7397     262.291793                                 251.654462
7773     267.212864                                 266.849270
1239     219.749815                                 226.555836
8072     200.341786                                 187.681363
1247     219.458958                                 228.378539
1079     225.209442                                 233.826684
```

```
#For the test data
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_absolute_percentage_error,r2_score

y_pred = lasso_mod.predict(x_test_log)

# Calculate MAE
mae_test = mean_absolute_error(y_test, y_pred)

# Calculate MSE
mse_test = mean_squared_error(y_test, y_pred)

# Calculate RMSE
rmse_test = np.sqrt(mse_test)
```

```
#calculate MAPE
mape_test = mean_absolute_percentage_error(y_test,y_pred)
#calculate R squared
rsq = r2_score(y_test, y_pred)
# Print MAE, MSE,RMSE, R squared
print("Mean Absolute Error (MAE): ", mae_test)
print("Mean Squared Error (MSE): ", mse_test)
print("Root Mean Squared Error (RMSE): ", rmse_test)
print("Mean absolute percentage error: ", mape_test)
print("R squared: ", rsq)
```

```
    Mean Absolute Error (MAE):  8.255949208554767
    Mean Squared Error (MSE):  139.6476252955515
    Root Mean Squared Error (RMSE):  11.817259635615674
    Mean absolute percentage error:  0.034813327678556105
    R squared:  0.7800258393658079
```

```
!pip install nbconvert
```

```
    Requirement already satisfied: nbconvert in /usr/local/lib/python3.10/dist-packages (6.5.4)
    Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from nbconvert) (4.9.4)
    Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (4.12.3)
    Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from nbconvert) (6.1.0)
    Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-packages (from nbconvert) (0.7.1)
    Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (0.4)
    Requirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (3.1.4)
    Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (5.7.2)
    Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.10/dist-packages (from nbconvert) (0.3.0
    Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (2.1.5)
    Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (0.8.4)
    Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (0.10.0)
    Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (5.10.4)
    Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from nbconvert) (24.0)
    Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (1.5.1
    Requirement already satisfied: pygments>=2.4.1 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (2.16.1)
    Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (1.3.0)
    Requirement already satisfied: traitlets>=5.0 in /usr/local/lib/python3.10/dist-packages (from nbconvert) (5.7.1)
    Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dist-packages (from jupyter-core>=4.7->
    Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.10/dist-packages (from nbclient>=0.5.
    Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.10/dist-packages (from nbformat>=5.1->n
    Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.10/dist-packages (from nbformat>=5.1->nbconv
    Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->nbconve
```