```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, classification_report

# Assuming your data is stored in a CSV file named 'data.csv'
data = pd.read_csv('/content/btp_244.csv')

# Set a limit for the target variable
limit = 0.5

# Create a binary target variable
data['target_binary'] = (data['vonMises_(MPa)'] > limit).astype(int)

# Separate features and target variable
X = data.drop('vonMises_(MPa)', axis=1)
y = data['target_binary']

# Handle missing values in X
imputer = SimpleImputer(strategy='median')  # replace 'median' with 'mean' or 'most_frequent' as needed
X_imputed = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)


# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_imputed, y, test_size=0.2, random_state=42)

# Train logistic regression model
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# Make predictions
y_pred = logreg.predict(X_test)

# Calculate accuracy and classification report
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```
print('Classification Report:')
print(classification_report(y_test, y_pred))

# Train linear regression model
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train.apply(lambda x: 1 if x > 0 else 0))

# Calculate parameters for linear regression model
parameters = {'intercept': linear_reg.intercept_, 'coefficients': linear_reg.coef_[0]}
print(parameters)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
Accuracy: 0.9977964957173828
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        62
           1       1.00      1.00      1.00     28075

    accuracy                           1.00     28137
   macro avg       0.50      0.50      0.50     28137
weighted avg       1.00      1.00      1.00     28137

{'intercept': -7.771561172376096e-16, 'coefficients': -4.003901350263413e-20}
```

```
import statsmodels.api as sm

X = data[['vonMises_(MPa)', 'disp_(mm)_Magnitude']]
#y = data['y']
X = sm.add_constant(X)

model = sm.OLS(y, X, missing='drop')
model_result = model.fit()

print(model_result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          target_binary   R-squared:                       0.013
Model:                            OLS   Adj. R-squared:                  0.013
Method:                 Least Squares   F-statistic:                     925.0
Date:                Fri, 17 May 2024   Prob (F-statistic):               0.00
Time:                        09:52:36   Log-Likelihood:             2.3088e+05
No. Observations:              140681   AIC:                          -4.617e+05
Df Residuals:                  140678   BIC:                          -4.617e+05
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                     0.9848      0.000   3020.603      0.000       0.984       0.985
vonMises_(MPa)       2.731e-05   7.31e-07     37.389      0.000    2.59e-05    2.87e-05
disp_(mm)_Magnitude     0.0298      0.002     16.195      0.000       0.026       0.033
==============================================================================
Omnibus:                   309463.765   Durbin-Watson:                   0.018
Prob(Omnibus):                  0.000   Jarque-Bera (JB):       1101579827.418
Skew:                         -20.685   Prob(JB):                         0.00
Kurtosis:                     434.528   Cond. No.                     6.22e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.22e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
new_data = pd.read_csv('/content/btp_226.csv')
```

```
limit = 756
new_data['new_target_binary'] = (new_data['vonMises_(MPa)'] > limit).astype(int)

# Select 2 independent variables
X_new = new_data[['vonMises_(MPa)', 'disp_(mm)_Magnitude']]
y_new = new_data['new_target_binary']

# Handle missing values in X_new
X_new = pd.DataFrame(imputer.fit_transform(X_new), columns=X_new.columns)

# Fit the linear regression model on the new dataset
linear_reg.fit(X_new, y_new.apply(lambda x: 1 if x > 0 else 0))

# Make predictions on the new dataset using the predict method
new_predictions_linear_reg = linear_reg.predict(X_new)

# Calculate the parameters of the linear regression model
parameters_linear_reg = {'intercept': linear_reg.intercept_, 'coefficients': linear_reg.coef_[0]}

# Print the parameters of the linear regression model
print('Parameters of linear regression model for new dataset:', parameters_linear_reg)
```

```
Parameters of linear regression model for new dataset: {'intercept': -0.10915638224123159, 'coefficients': 0.000668919385449939}
```