```
import pandas as pd
import numpy as np
from scipy.signal import savgol_filter
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold, cross_val_score
import joblib



# Replace 'train.csv' with your actual training file name
train = pd.read_csv('hacktrain.csv')
test = pd.read_csv('hacktest.csv')



ndvi_cols = [col for col in train.columns if '_N' in col]



def denoise_series(row):
    if row.isnull().sum() > len(row) // 2:
        return row
    filled = row.interpolate(limit_direction='both')
    window = min(7, len(row) if len(row)%2==1 else len(row)-1)
    if window < 3:
        return filled
    filtered = savgol_filter(filled, window_length=window, polyorder=2)
    return pd.Series(filtered, index=row.index)

X_train_denoised = train[ndvi_cols].apply(denoise_series, axis=1)
X_test_denoised = test[ndvi_cols].apply(denoise_series, axis=1)



imputer = KNNImputer(n_neighbors=3)
X_train_imputed = pd.DataFrame(imputer.fit_transform(X_train_denoised), columns=ndvi_cols)
X_test_imputed = pd.DataFrame(imputer.transform(X_test_denoised), columns=ndvi_cols)



def extract_features(row):
    ndvi = np.array(row)
    return pd.Series({
        'ndvi_mean': np.mean(ndvi),
        'ndvi_std': np.std(ndvi),
        'ndvi_min': np.min(ndvi),
        'ndvi_max': np.max(ndvi),
        'ndvi_range': np.max(ndvi) - np.min(ndvi),
        'ndvi_argmax': np.argmax(ndvi),
        'ndvi_argmin': np.argmin(ndvi),
        'ndvi_median': np.median(ndvi),
        'ndvi_q25': np.percentile(ndvi, 25),
        'ndvi_q75': np.percentile(ndvi, 75),
        'ndvi_skew': pd.Series(ndvi).skew(),
    })

X_train_features = X_train_imputed.apply(extract_features, axis=1)
X_test_features = X_test_imputed.apply(extract_features, axis=1)



# Replace 'class' with the actual name of the label column if it's different
le = LabelEncoder()
y = le.fit_transform(train['class'].astype(str))



scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_features)
X_test_scaled = scaler.transform(X_test_features)



clf = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=1000, class_weight='balanced', random_state=42)
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
scores = cross_val_score(clf, X_train_scaled, y, cv=cv, scoring='accuracy')
print(f'CV Accuracy: {np.mean(scores):.4f} ± {np.std(scores):.4f}')
clf.fit(X_train_scaled, y)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in versi
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in versi
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in versi
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in versi
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in versi
  warnings.warn(
CV Accuracy: 0.4785 ± 0.0102
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in versi
  warnings.warn(
```

```
LogisticRegression

LogisticRegression(class_weight='balanced', max_iter=1000,
                   multi_class='multinomial', random_state=42)
```

```
joblib.dump(clf, 'logreg_model.pkl')
joblib.dump(le, 'label_encoder.pkl')
joblib.dump(scaler, 'scaler.pkl')
```

```
['scaler.pkl']
```

```
y_test_pred = clf.predict(X_test_scaled)
y_test_labels = le.inverse_transform(y_test_pred)
```

```
submission = pd.DataFrame({
    'ID': test['ID'],
    'class': y_test_labels
})
submission.to_csv('submission.csv', index=False)
print('submission.csv created!')
```

```
submission.csv created!
```