```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load the dataset
data = pd.read_csv('/content/creditcard.csv')

# Separate features and target
X = data.drop('Class', axis=1)
y = data['Class']

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

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

# Load the dataset
data = pd.read_csv('creditcard.csv')

# Check for NaN values
print(data.isnull().sum())

# Remove rows with NaN values
data = data.dropna()

# Separate features and target
X = data.drop('Class', axis=1)
y = data['Class']

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

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize the model
model = LogisticRegression()

# Train the model
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Time      0
V1        0
V2        0
V3        1
V4        1
V5        1
V6        1
V7        1
V8        1
V9        1
V10       1
V11       1
V12       1
V13       1
V14       1
V15       1
V16       1
V17       1
V18       1
V19       1
V20       1
V21       1
V22       1
```

```
V23       1
V24       1
V25       1
V26       1
V27       1
V28       1
Amount    1
Class     1
dtype: int64
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      8308
         1.0       0.88      0.52      0.65        29

    accuracy                           1.00      8337
   macro avg       0.94      0.76      0.83      8337
weighted avg       1.00      1.00      1.00      8337

Accuracy: 0.9980808444284515
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status
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(
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
import ipywidgets as widgets
from IPython.display import display
import numpy as np

# Load and preprocess the dataset
data = pd.read_csv('creditcard.csv')

# Remove rows with NaN values
data = data.dropna()
X = data.drop('Class', axis=1)
y = data['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train the model
model = LogisticRegression()
model.fit(X_train, y_train)

# Create widgets for all features
input_widgets = {}
for column in X.columns:
    input_widgets[column] = widgets.FloatText(description=column)

# Function to handle NaNs and make a prediction
def predict_fraud(change):
    input_data = pd.DataFrame({col: [widget.value] for col, widget in input_widgets.items()})
    # Fill NaNs with the mean of the respective column
    input_data.fillna(X.mean(), inplace=True)
    input_data = scaler.transform(input_data)
    prediction = model.predict(input_data)
    result.value = 'Fraud' if prediction[0] == 1 else 'Not Fraud'

# Add observers for all features
for widget in input_widgets.values():
    widget.observe(predict_fraud, names='value')

# Display the widgets
for widget in input_widgets.values():
    display(widget)

# Create a widget to display the result
result = widgets.Text(value='', description='Result', disabled=True)
display(result)

# Sample data to trigger a fraud prediction
sample_data = {
    'Time': 0,
    'V1': -3.043539,
```

```
        'V2': -4.947317,
        'V3': 1.304961,
        'V4': -4.015027,
        'V5': -5.536856,
        'V6': -2.830055,
        'V7': -2.555244,
        'V8': 1.480740,
        'V9': 4.093916,
        'V10': -4.603275,
        'V11': -2.474320,
        'V12': -4.545878,
        'V13': -0.363930,
        'V14': -1.597490,
        'V15': -3.338337,
        'V16': -4.345535,
        'V17': 1.405946,
        'V18': 2.926620,
        'V19': -2.403825,
        'V20': -2.621535,
        'V21': 1.543790,
        'V22': -1.357746,
        'V23': -1.894934,
        'V24': -2.727046,
        'V25': 2.456949,
        'V26': -0.304241,
        'V27': 0.676067,
        'V28': 1.965775,
        'Amount': 500.0
    }

    # Set the widgets to sample data values to trigger the prediction
    for key, value in sample_data.items():
        input_widgets[key].value = value
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (sta
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(
```

Time	0
V1	-3.043539
V2	-4.947317
V3	1.304961
V4	-4.015027
V5	-5.536856
V6	-2.830055
V7	-2.555244
V8	1.48074
V9	4.093916
V10	-4.603275
V11	-2.47432
V12	-4.545878
V13	-0.36393
V14	-1.59749
V15	-3.338337
V16	-4.345535
V17	1.405946
V18	2.92662
V19	-2.403825
V20	-2.621535
V21	1.54379
V22	-1.357746
V23	-1.894934
V24	-2.727046
V25	2.456949
V26	-0.304241
V27	0.676067
V28	1.965775
Amount	500
Result	Not Fraud

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
import ipywidgets as widgets
from IPython.display import display

# Load and preprocess the dataset
data = pd.read_csv('creditcard.csv')
# Remove rows with NaN values
data = data.dropna()
X = data.drop('Class', axis=1)
y = data['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train the model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Create widgets for all features
```

```
# Create widgets for all features
input_widgets = {}
for column in X.columns:
    input_widgets[column] = widgets.FloatText(description=column)

# Function to handle NaNs and make a prediction
def predict_fraud(change):
    input_data = pd.DataFrame({col: [widget.value] for col, widget in input_widgets.items()})
    # Fill NaNs with the mean of the respective column
    input_data.fillna(X.mean(), inplace=True)
    # Preprocess input data using the same scaler
    input_data_scaled = scaler.transform(input_data)
    prediction = model.predict(input_data_scaled)
    result.value = 'Fraud' if prediction[0] == 1 else 'Not Fraud'

# Add observers for all features
for widget in input_widgets.values():
    widget.observe(predict_fraud, names='value')

# Display the widgets
for widget in input_widgets.values():
    display(widget)

# Create a widget to display the result
result = widgets.Text(value='', description='Result', disabled=True)
display(result)

# Example fraudulent transaction data
fraud_sample_data = {
}

# Set the widgets to sample data values to trigger the prediction
for key, value in fraud_sample_data.items():
    input_widgets[key].value = value
predict_fraud(None)  # Trigger prediction immediately after setting sample data
```

Time	19
V1	13
V2	13
V3	12
V4	14
V5	12
V6	17
V7	14
V8	14
V9	10
V10	10
V11	7
V12	12
V13	8
V14	10