

```
[23]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix
```

```
[24]: # Specify the file path
file = "D:\\intership\\New folder\\Credit Card Fraud Detection\\creditcard.csv"

# Read the CSV file into a DataFrame
data = pd.read_csv(file)

# Display the first few rows of the dataset
data.head()
```

```
[24]:
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V2
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.18911
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.12588
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.13908
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.22192
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.50229

5 rows × 31 columns

```
[6]: # Check missing values
data.isnull().sum()
```

```
[6]: Time      0
V1          0
V2          0
V3          0
V4          0
V5          0
V6          0
V7          0
V8          0
V9          0
V10         0
V11         0
V12         0
V13         0
V14         0
V15         0
V16         0
V17         0
V18         0
V19         0
V20         0
V21         0
V22         0
V23         0
V24         0
V25         0
V26         0
V27         0
V28         0
Amount      0
Class       0
dtype: int64
```

```
[7]: # Drop missing values
data = data.dropna()
data.head()
```

```
[7]:
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V2
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.18911
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.12588
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.13908
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.22192
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.50229

5 rows × 31 columns

```
[25]: # Separate features and target variable
X = data.drop(columns=['Class'])
y = data['Class']

# Normalize the features using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
[26]: # Apply oversampling to address class imbalance
oversampler = RandomOverSampler()
X_resampled, y_resampled = oversampler.fit_resample(X_scaled, y)
```

```
[27]: # Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)
```

```
[30]: # Initialize and train a Logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

C:\Users\Admin\anaconda3\lib\site-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](https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)  
n\_iter\_i = \_check\_optimize\_result(

```
[30]: LogisticRegression
LogisticRegression()
```

```
[31]: # Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model using precision, recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print evaluation metrics
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("Confusion Matrix:")
print(conf_matrix)
```

Precision: 0.9649122807017544  
Recall: 0.9016393442622951  
F1-score: 0.9322033898305084  
Confusion Matrix:  
[[80 2]  
 [ 6 55]]

```
•[35]: # Apply undersampling to further balance the classes
undersampler = RandomUnderSampler()
X_resampled, y_resampled = undersampler.fit_resample(X_train, y_train)

# Retrain the model using resampled data
model.fit(X_resampled, y_resampled)

# Evaluate refined model
y_pred = model.predict(X_test)

# Evaluate the refined model using the same metrics as before
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Precision (Refined Model - Undersampling):", precision)
print("Recall (Refined Model - Undersampling):", recall)
print("F1-score (Refined Model - Undersampling):", f1)
print("Confusion Matrix (Refined Model - Undersampling):")
print(conf_matrix)
```

Precision (Refined Model - Undersampling): 0.9649122807017544  
Recall (Refined Model - Undersampling): 0.9016393442622951  
F1-score (Refined Model - Undersampling): 0.9322033898305084  
Confusion Matrix (Refined Model - Undersampling):  
[[80 2]  
 [ 6 55]]

```
[ ]:
```

```
[ ]:
```