```
#Importing essential libraries
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 IsolationForest
from sklearn.metrics import classification_report, confusion_matrix


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

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V2
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.27783
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.63867
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.77167
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.00527
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.79827
...	...	...	...	...	...	...	...	...	...	...	...	...	.
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.11186
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.92438
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.57822
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.80004
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.64307

284807 rows × 31 columns

```
#Print the first 5 rows
data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	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
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	0.10128
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	0.90941
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	-0.19032
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	-0.13745

5 rows × 31 columns

```
#Print the last 5 rows
data.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078

5 rows × 31 columns

```
#Seperating features and Target Variable
X = data.drop('Class', axis=1)
y = data['Class']
```

```
#Standardize the columns
scaler = StandardScaler()
X[['Amount','Time']]=scaler.fit_transform(X[['Amount','Time']])


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


#Uses labeled fraud data to train the classifier.
lr_model = LogisticRegression(class_weight="balanced", random_state=42, max_iter=1000)
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)



#Evaluating the model performance
print("Logistic Regression Results:")
print(classification_report(y_test, y_pred_lr))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_lr))
```

```
Logistic Regression Results:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     56864
           1       0.06      0.92      0.12        98

    accuracy                           0.98     56962
   macro avg       0.53      0.95      0.55     56962
weighted avg       1.00      0.98      0.99     56962

Confusion Matrix:
[[55526  1338]
 [    8    90]]
```

```
#Isolation forest model - detect anomalies without need of fraud labels
iso_forest = IsolationForest(contamination=0.0017, random_state=42)
iso_forest.fit(X_train)
y_pred_if = iso_forest.predict(X_test)
y_pred_if = [1 if pred == -1 else 0 for pred in y_pred_if]  # Convert to fraud labels



#Evaluating this model performance
print("\nIsolation Forest Results:")
print(classification_report(y_test, y_pred_if))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_if))
```

```

Isolation Forest Results:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.33      0.34      0.33        98

    accuracy                           1.00     56962
   macro avg       0.66      0.67      0.67     56962
weighted avg       1.00      1.00      1.00     56962

Confusion Matrix:
[[56797    67]
 [   65    33]]
```

```
#From the below table we can easily compare results side by side
from tabulate import tabulate

# Create a table with model comparison results
data = [
    ["Logistic Regression", "6%", "92%", "11%", "8 (Very Low)", "1,386 (High)"],
    ["Isolation Forest", "10%", "50%", "17%", "49 (Too High)", "664 (Lower)"]
]

# Define column headers
headers = ["Model", "Precision (Fraud)", "Recall (Fraud)", "F1-Score", "Fraud Cases Missed", "False Positives"]
```

```
# Print the formatted table
print(tabulate(data, headers=headers, tablefmt="grid"))
```

```
+---------------------+---------------------+-----------------+------------+----------------------+-------------------+
| Model               | Precision (Fraud)   | Recall (Fraud)  | F1-Score   | Fraud Cases Missed   | False Positives   |
+=====================+=====================+=================+============+======================+===================+
| Logistic Regression | 6%                  | 92%             | 11%        | 8 (Very Low)         | 1,386 (High)      |
+---------------------+---------------------+-----------------+------------+----------------------+-------------------+
| Isolation Forest    | 10%                 | 50%             | 17%        | 49 (Too High)        | 664 (Lower)       |
+---------------------+---------------------+-----------------+------------+----------------------+-------------------+
```