Importing Libraries

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

In [7]:

```
data = pd.read_csv("C:\\Users\\ASHISH TIWARI\\Downloads\\Credit_Card_Fraud_Detection.csv")
data
```

Out[7]:

	Unnamed: 0	Customer_ID	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_10	A_11	A_12	A_13	A_14
0	0	15776156	1	22.08	11.460	2	4	4	1.585	0	0	0	1	2	100	1213
1	1	15739548	0	22.67	7.000	2	8	4	0.165	0	0	0	0	2	160	1
2	2	15662854	0	29.58	1.750	1	4	4	1.250	0	0	0	1	2	280	1
3	3	15687688	0	21.67	11.500	1	5	3	0.000	1	1	11	1	2	0	1
4	4	15715750	1	20.17	8.170	2	6	4	1.960	1	1	14	0	2	60	159
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
685	685	15808223	1	31.57	10.500	2	14	4	6.500	1	0	0	0	2	0	1
686	686	15769980	1	20.67	0.415	2	8	4	0.125	0	0	0	0	2	0	45
687	687	15675450	0	18.83	9.540	2	6	4	0.085	1	0	0	0	2	100	1
688	688	15776494	0	27.42	14.500	2	14	8	3.085	1	1	1	0	2	120	12
689	689	15592412	1	41.00	0.040	2	10	4	0.040	0	1	1	0	1	560	1

690 rows × 17 columns

In [8]:

```
data.describe()
```

Out[8]:

	Unnamed: 0	Customer_ID	A_1	A_2	A_3	A_4	A_5	A_6	A_7
count	690.000000	6.900000e+02	690.000000	690.000000	690.000000	690.000000	690.000000	690.000000	690.000000
mean	344.500000	1.569047e+07	0.678261	31.568203	4.758725	1.766667	7.372464	4.692754	2.223406
std	199.330128	7.150647e+04	0.467482	11.853273	4.978163	0.430063	3.683265	1.992316	3.346513
min	0.000000	1.556571e+07	0.000000	13.750000	0.000000	1.000000	1.000000	1.000000	0.000000
25%	172.250000	1.563169e+07	0.000000	22.670000	1.000000	2.000000	4.000000	4.000000	0.165000
50%	344.500000	1.569016e+07	1.000000	28.625000	2.750000	2.000000	8.000000	4.000000	1.000000
75%	516.750000	1.575190e+07	1.000000	37.707500	7.207500	2.000000	10.000000	5.000000	2.625000
max	689.000000	1.581544e+07	1.000000	80.250000	28.000000	3.000000	14.000000	9.000000	28.500000

In [10]:

```
data.shape
```

Out[10]:

```
(690, 17)
```

In [11]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 690 entries, 0 to 689
Data columns (total 17 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Unnamed: 0   690 non-null    int64  
 1   Customer_ID  690 non-null    int64  
 2   A_1          690 non-null    int64  
 3   A_2          690 non-null    float64
 4   A_3          690 non-null    float64
 5   A_4          690 non-null    int64  
 6   A_5          690 non-null    int64  
 7   A_6          690 non-null    int64  
 8   A_7          690 non-null    float64
 9   A_8          690 non-null    int64  
 10  A_9          690 non-null    int64  
 11  A_10         690 non-null    int64  
 12  A_11         690 non-null    int64  
 13  A_12         690 non-null    int64  
 14  A_13         690 non-null    int64  
 15  A_14         690 non-null    int64  
 16  class        690 non-null    int64  
dtypes: float64(3), int64(14)
memory usage: 91.8 KB
```

In [12]:

```
data.isnull().sum()
```

Out[12]:

```
Unnamed: 0     0
Customer_ID    0
A_1            0
A_2            0
A_3            0
A_4            0
A_5            0
A_6            0
A_7            0
A_8            0
A_9            0
A_10           0
A_11           0
A_12           0
A_13           0
A_14           0
class          0
dtype: int64
```

In [14]:

```
data['class'].value_counts()
```

Out[14]:

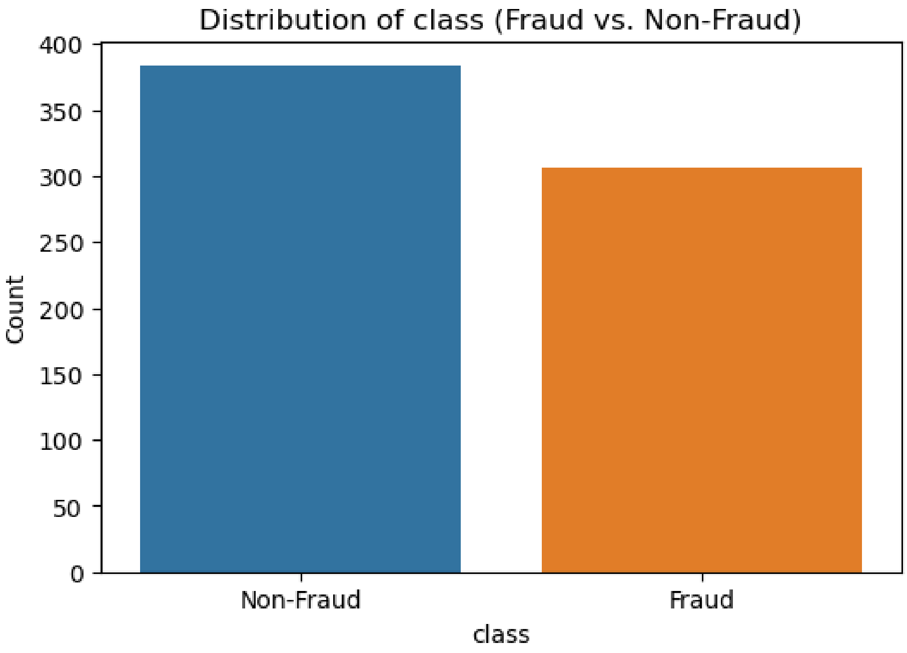
```
class
0    383
1    307
Name: count, dtype: int64
```

Data Visualization

In [16]:

```
# Custom labels for the classes
class_labels = {0: 'Non-Fraud', 1: 'Fraud'}

# Distribution of Class (Fraud vs. Non-Fraud)
plt.figure(figsize=(6, 4))
sns.countplot(data=data, x='class')
plt.title('Distribution of class (Fraud vs. Non-Fraud)')
plt.xlabel('class')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=[class_labels[0], class_labels[1]])  # Custom labels for the x-axis
plt.show()
```



Separating Features and Target

In [32]:

```
x = data.drop('class', axis=1)
y = data[['class']]
```

In [33]:

```
x.head()
```

Out[33]:

	Unnamed: 0	Customer_ID	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_10	A_11	A_12	A_13	A_14
0	0	15776156	1	22.08	11.46	2	4	4	1.585	0	0	0	1	2	100	1213
1	1	15739548	0	22.67	7.00	2	8	4	0.165	0	0	0	0	2	160	1
2	2	15662854	0	29.58	1.75	1	4	4	1.250	0	0	0	1	2	280	1
3	3	15687688	0	21.67	11.50	1	5	3	0.000	1	1	11	1	2	0	1
4	4	15715750	1	20.17	8.17	2	6	4	1.960	1	1	14	0	2	60	159

In [34]:

```
y.head()
```

Out[34]:

	class
0	0
1	0
2	0
3	1
4	1

In [35]:

```
print(x.shape)
print(y.shape)
```

```
(690, 16)
(690, 1)
```

Splitting Data Into Train and Test

In [37]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=51)
```

In [38]:

```
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(552, 16) (138, 16) (552, 1) (138, 1)
```

Model Training

In [43]:

```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression()
log.fit(x_train,y_train)
```

```
C:\ANACONDA\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for e
xample using ravel().
  y = column_or_1d(y, warn=True)
```

Out[43]:

```
▾ LogisticRegression
LogisticRegression()
```

In [46]:

```
y_pred1 = log.predict(x_test)
```

In [47]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred1)
```

Out[47]:

```
0.6666666666666666
```

In [50]:

```
rf = RandomForestClassifier()
rf.fit(x_train, y_train)
```

```
C:\ANACONDA\Lib\site-packages\sklearn\base.py:1151: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using
ravel().
  return fit_method(estimator, *args, **kwargs)
```

Out[50]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [53]:

```
y_pred2 = rf.predict(x_test)
accuracy_score(y_test, y_pred2)
```

Out[53]:

```
0.8913043478260869
```

Model Evaluation

In [54]:

```
print("Logistic Regression Performance:")
print(classification_report(y_test, y_pred1))
print()
print("Random Forest Classifier Performance:")
print(classification_report(y_test, y_pred2))
```

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

           0       0.62      0.95      0.75        73
           1       0.85      0.35      0.50        65

    accuracy                           0.67       138
   macro avg       0.74      0.65      0.62       138
weighted avg       0.73      0.67      0.63       138


Random Forest Classifier Performance:
              precision    recall  f1-score   support

           0       0.90      0.89      0.90        73
           1       0.88      0.89      0.89        65

    accuracy                           0.89       138
   macro avg       0.89      0.89      0.89       138
weighted avg       0.89      0.89      0.89       138
```