Financial Fraud Prediction

In [60]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report
```

In [31]:
```
p1= pd.read_csv('creditcard.csv')
```

In [32]:
```
p1
```

Out[32]:

	Time	V1	V2	V3	V4	V5	V6	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.2395
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.0788
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.7914
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.2376
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.5929
...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.9182
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.0243
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.2968
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.6861
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.5770

284807 rows × 31 columns

In [33]:
```
p1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

Checking for null values

In [35]:
```
p1.isnull().sum()
```

Out[35]:
```
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
```

Total Number of legit and fraud transactions

In [41]:
```
legit_count = len(p1[p1['Class'] == 0])
fraud_count = len(p1[p1['Class'] == 1])
print (f"Normal transactions: {legit_count}")
print (f"Fraud cases: {fraud_count}")
```

```
Normal transactions: 284315
Fraud cases: 492
```

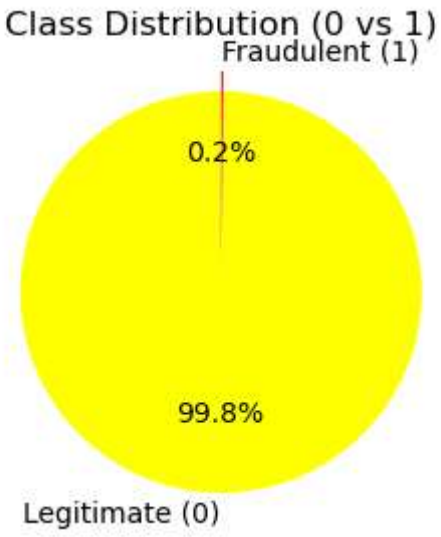
Creating pie chart for the fradulant and normal transaction

In [44]:
```
class_counts = p1['Class'].value_counts()

# Create pie chart

plt.figure(figsize=(5, 3))
plt.pie(class_counts,
```

```
labels=['Legitimate (0)', 'Fraudulent (1)'],
colors=['yellow', 'red'],
autopct='%1.1f%%',
startangle=90,
explode=(0, 0.1))
plt.title('Class Distribution (0 vs 1)')
plt.axis('equal')
plt.show()
```



Prediction

In [46]:
```
X = p1.drop(['Class', 'Time'], axis=1)
y = p1['Class']
```

In [47]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Test, train and split

In [50]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.1,
 random_state=42, stratify=y)
```

Create the Random Forest classification model as a result of high imbalance between fradulant(0.2%) and legitmate (99.8%) transactions

In [61]:
```
rf_model = RandomForestClassifier(n_estimators=100, class_weight='balanced', random
```

In [62]:
```
rf_model
```

Out[62]:
```
RandomForestClassifier(class_weight='balanced', random_state=42)
```

Fitting training data into the model

In [66]:
```
rf_model.fit(X_train, y_train)
```

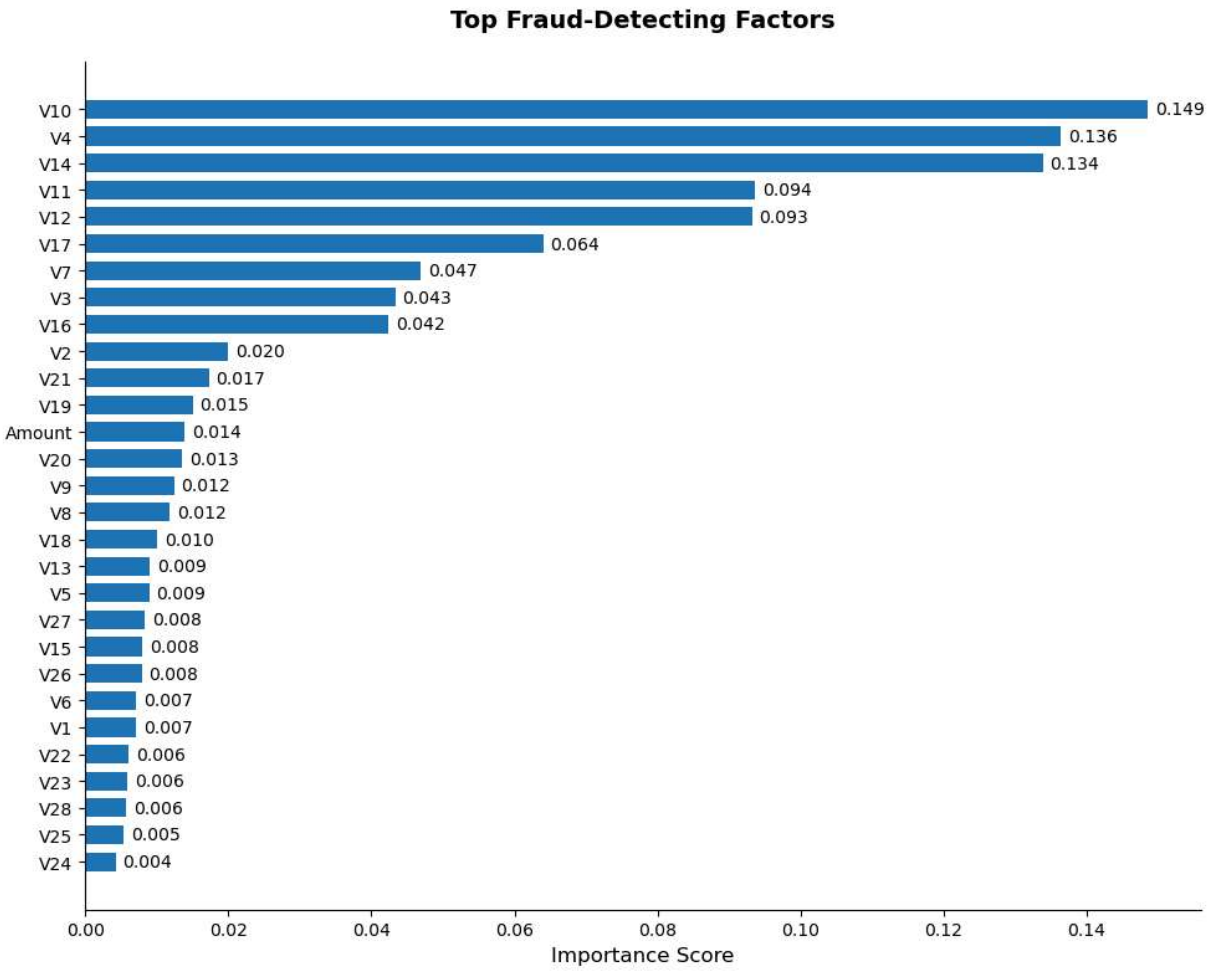
Out[66]:
```
RandomForestClassifier(class_weight='balanced', random_state=42)
```

Prediction and evaluation

In [ ]:
```
y_pred = rf_model.predict(X_test)
```

In [ ]:
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, digits=4))
```

In [72]:
```
 # Get and sort feature importances
 importances = rf_model.feature_importances_
 sorted_idx = importances.argsort()
 plt.figure(figsize=(10, 8))
 plt.barh(X.columns[sorted_idx], importances[sorted_idx],
 color='#1f77b4', edgecolor='none', height=0.7)
 plt.title('Top Fraud-Detecting Factors', pad=20, fontsize=14, fontweight='bold')
 plt.xlabel('Importance Score', fontsize=12)
 plt.gca().spines['top'].set_visible(False)
 plt.gca().spines['right'].set_visible(False)
# Add value labels
 for i, v in enumerate(importances[sorted_idx]):
     plt.text (v + 0.001, i, f"{v:.3f}", color='black', ha='left', va='center', fon
 plt.tight_layout()
 plt.show()
```


Top Fraud-Detecting Factors
V10 0.149
V4 0.136
V14 0.134
V11 0.094
V12 0.093
V17 0.064
V7 0.047
V3 0.043
V16 0.042
V2 0.020
V21 0.017
V19 0.015
Amount 0.014
V20 0.013
V9 0.012
V8 0.012
V18 0.010
V13 0.009
V5 0.009
V27 0.008
V15 0.008
V26 0.008
V6 0.007
V1 0.007
V22 0.006
V23 0.006
V28 0.006
V25 0.005
V24 0.004
0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14
Importance Score

In [ ]: