全球洗钱案件数据分析

Time：24小时制时间

Date：日期

Sender_account：发送方账户

Receiver_account：接收方账户

Amount：金额

Payment_currency：支付货币

Received_currency：收款货币

Sender_bank_location：汇款银行地点

Receiver_bank_location：收款银行地点

Payment_type：支付类型

Is_laundering：是否洗钱

Laundering_type：洗钱类型

数据来源：IEEE《加强反洗钱：开发综合交易监控数据集》https://ieeexplore.ieee.org/document/10356193 (https://ieeexplore.ieee.org/document/10356193) 该数据集包含 12 个特征和 28 个类型（分为11个正常和17个可疑）。这些是根据现有数据集、学术文献和对反洗钱专家的采访选择的。同时，我通过随机抽样抽取其中10万数据量的数据集作用于本次实验 分析目的：洗钱仍然是一个重大的全球性问题，保护财产安全成为一个重要任务。通过本次实验进行数据分析，展示该数据集所表达的内容并且尝试挖掘出其潜在的价值

一、导入必要模块、库并读取数据

In [1]:
```
#导入所需要的库
import pandas as pd
import numpy as np
from pyecharts import options as opts
from pyecharts.charts import Line,Bar,Scatter,Pie,Map
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
#读取数据
df = pd.read_csv('saml-d_副本_10w.csv')
df.head()
```

Out[2]:

	Time	Date	Sender_account	Receiver_account	Amount	Payment_currency	Received_currency	Sender_bank_location	Receiver_bank_locati
0	10:40:26	2022-10-07	3776976361	8774487578	6713.21	UK pounds	UK pounds	UK	U
1	10:40:31	2022-10-07	6626359609	9024705045	547.39	UK pounds	UK pounds	UK	U
2	10:41:21	2022-10-07	9734083970	4117514495	4863.13	UK pounds	UK pounds	UK	U
3	10:46:37	2022-10-07	7401327478	4336451277	2603.30	UK pounds	UK pounds	UK	U
4	10:46:58	2022-10-07	5959325606	4054367503	546.33	UK pounds	UK pounds	UK	U

二、查看数据基本情况

In [3]:
```
#查看数据大小
df.shape
```

Out[3]: (104841, 12)

In [4]:
```
#查看特征列
df.columns.tolist()
```

Out[4]:
```
['Time',
 'Date',
 'Sender_account',
 'Receiver_account',
 'Amount',
 'Payment_currency',
 'Received_currency',
 'Sender_bank_location',
 'Receiver_bank_location',
 'Payment_type',
 'Is_laundering',
 'Laundering_type']
```

In [5]:
```
#查看数据信息
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104841 entries, 0 to 104840
Data columns (total 12 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   Time                   104841 non-null  object
 1   Date                   104841 non-null  object
 2   Sender_account         104841 non-null  int64
 3   Receiver_account       104841 non-null  int64
 4   Amount                 104841 non-null  float64
 5   Payment_currency       104841 non-null  object
 6   Received_currency      104841 non-null  object
 7   Sender_bank_location   104841 non-null  object
 8   Receiver_bank_location 104841 non-null  object
 9   Payment_type           104841 non-null  object
 10  Is_laundering          104841 non-null  int64
 11  Laundering_type        104841 non-null  object
dtypes: float64(1), int64(3), object(8)
memory usage: 9.6+ MB
```

In [6]:
```
#查看数据值统计情况
df.describe()
```

Out[6]:

	Sender_account	Receiver_account	Amount	Is_laundering
count	1.048410e+05	1.048410e+05	1.048410e+05	104841.000000
mean	5.034716e+09	5.000879e+09	1.173824e+04	0.094171
std	2.881259e+09	2.871925e+09	1.305849e+05	0.292068
min	9.217200e+04	9.217200e+04	4.230000e+00	0.000000
25%	2.550374e+09	2.528353e+09	2.204980e+03	0.000000
50%	5.038834e+09	4.978779e+09	6.051090e+03	0.000000
75%	7.531251e+09	7.483116e+09	1.044702e+04	0.000000
max	9.999913e+09	9.999804e+09	1.261850e+07	1.000000

三、特征工程

In [7]:
```
#查看数据是否有缺失值
df.isnull().sum()
```

Out[7]:
```
Time                    0
Date                    0
Sender_account          0
Receiver_account        0
Amount                  0
Payment_currency        0
Received_currency       0
Sender_bank_location    0
Receiver_bank_location  0
Payment_type            0
Is_laundering           0
Laundering_type         0
dtype: int64
```

In [8]:
```
#查看数据是否有重复值
df.duplicated().any()
```

Out[8]:
```
False
```

In [9]:
```
#数据类型转换
df['DateTime'] = pd.to_datetime(df['Date'] + ' ' + df['Time'])
df.drop(columns=['Date','Time'],inplace=True)
```

In [10]:
```
#是否洗钱统计
df.groupby('Is_laundering').size().sort_values(ascending=False)
```

Out[10]:
```
Is_laundering
0    94968
1     9873
dtype: int64
```

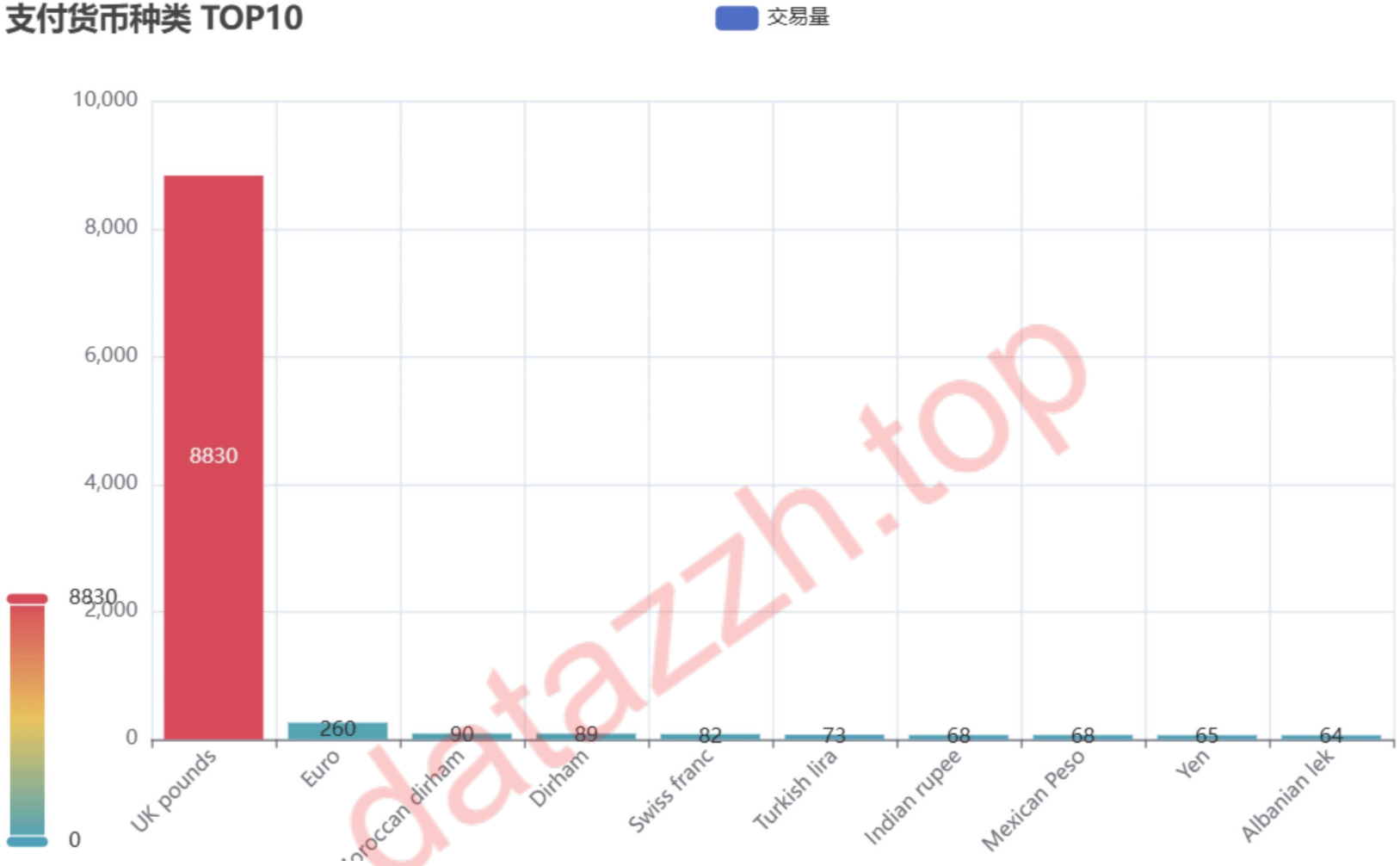
In [11]:
```
# 获取只有被洗钱的数据
df_laundering = df[df['Is_laundering'] == 1]
df_laundering = df_laundering.copy()
```

四、pyecharts数据可视化

图一：支付货币种类 TOP10

In [12]:
```
# 计算支付货币种类前10数据量
Payment_currency_top10 = df_laundering.groupby('Payment_currency').size().sort_values(ascending=False).head(10)
bar1 = (
    Bar()
    .add_xaxis(Payment_currency_top10.index.tolist())
    .add_yaxis('交易量',Payment_currency_top10.values.tolist())
    .set_global_opts(
        title_opts=opts.TitleOpts(title='支付货币种类 TOP10'),
        xaxis_opts=opts.AxisOpts(axislabel_opts=opts.LabelOpts(rotate=45)),
        visualmap_opts=opts.VisualMapOpts(max_=8830)
    )
)
bar1.render_notebook()
```

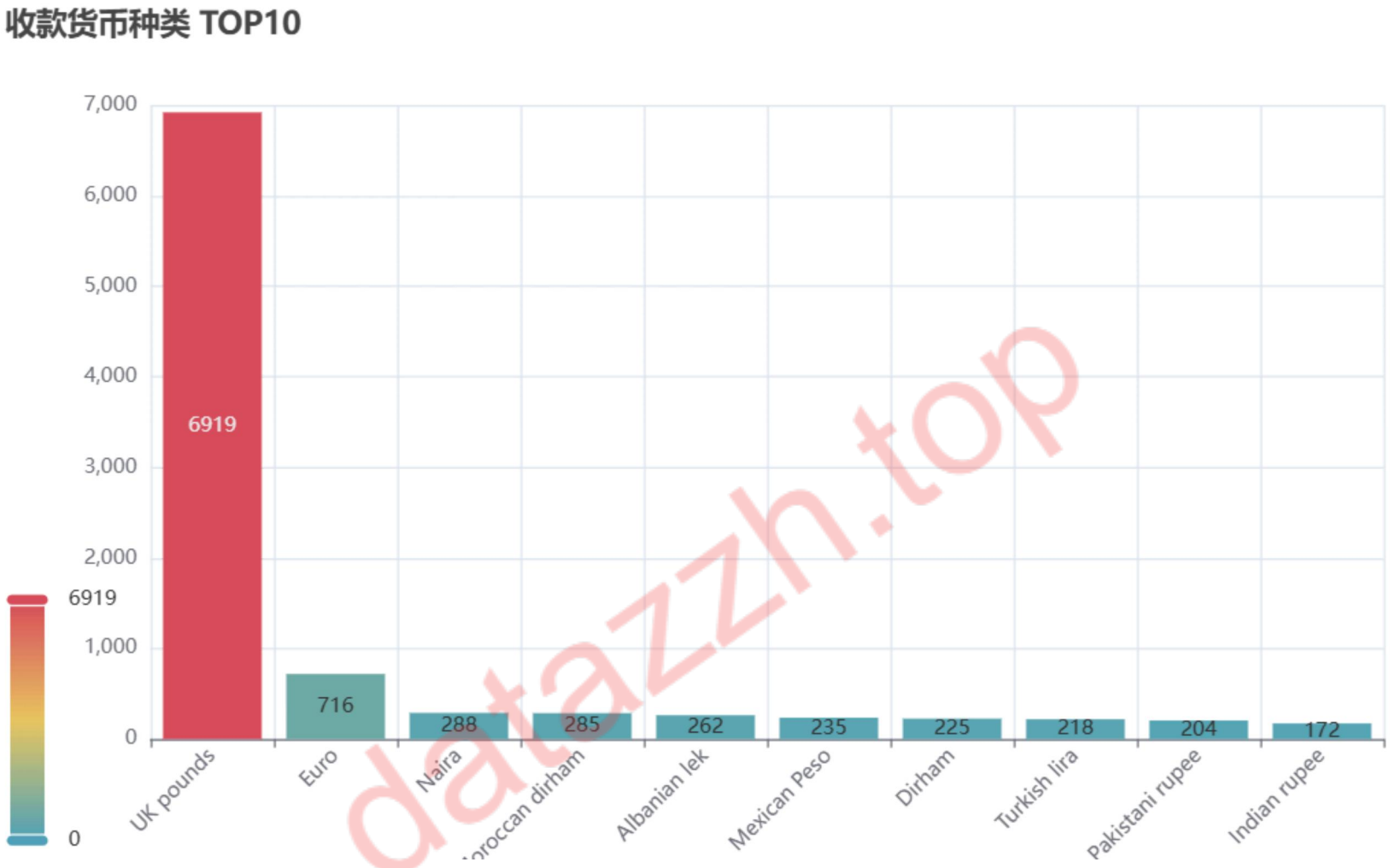
Out[12]:



图二：收款货币种类 TOP10

In [13]:
```
# 计算收款货币种类前10数据量
Received_currency_top10 = df_laundering.groupby('Received_currency').size().sort_values(ascending=False).head(10)
bar2 = (
    Bar()
    .add_xaxis(Received_currency_top10.index.tolist())
    .add_yaxis('',Received_currency_top10.values.tolist())
    .set_global_opts(
        title_opts=opts.TitleOpts(title='收款货币种类 TOP10'),
        xaxis_opts=opts.AxisOpts(axislabel_opts=opts.LabelOpts(rotate=45)),
        visualmap_opts=opts.VisualMapOpts(max_=6919)
    )
)
bar2.render_notebook()
```

Out[13]:



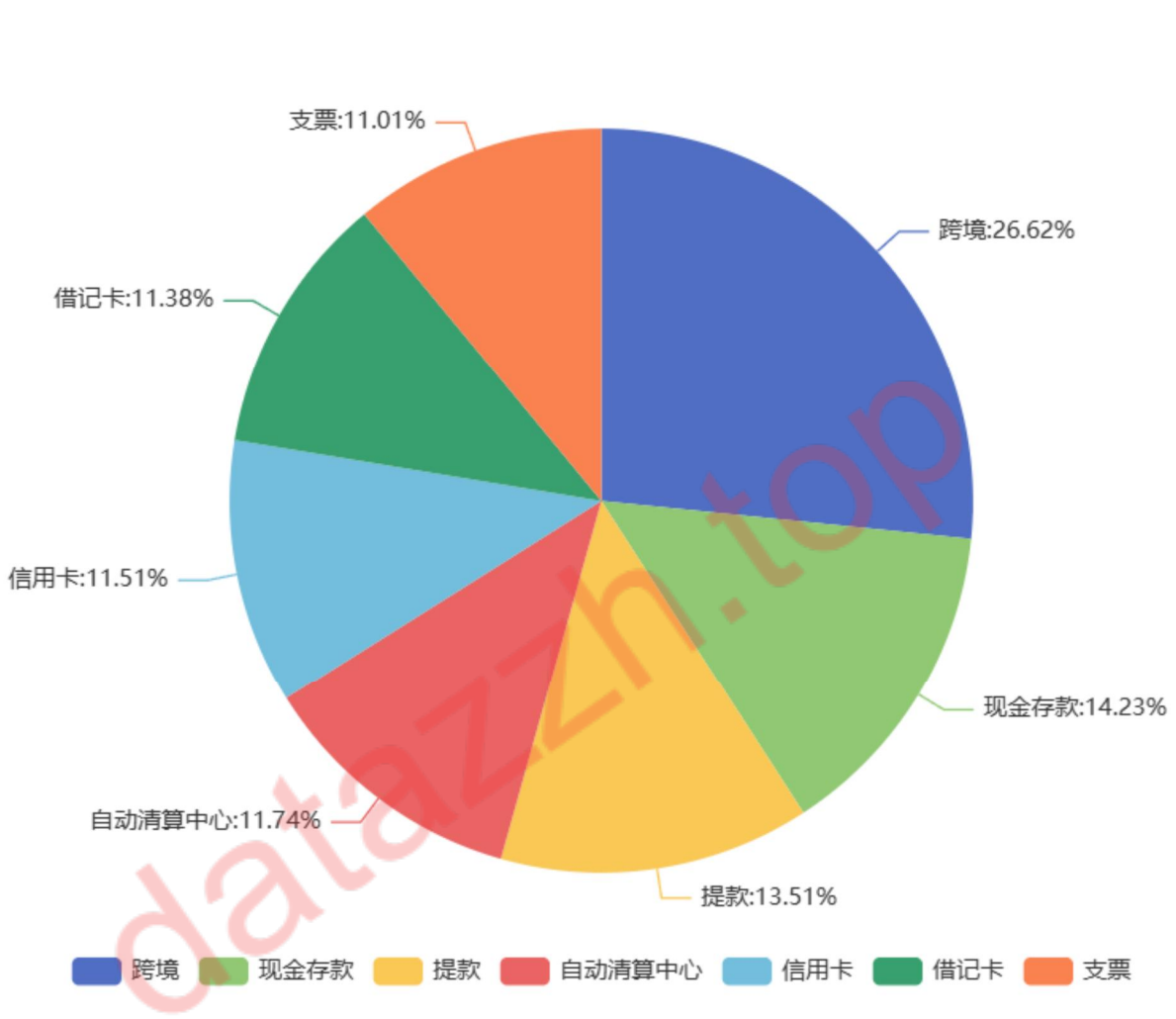
通过支付和汇款货币TOP10的柱状图可知，涉及洗钱案件使用最多的货币为英镑

图三：Top10 支付类型

In [14]:
```
# 为更直观展示数据，对支付类型进行中文映射翻译
Payment_type_cn = {
    'Credit card':'信用卡',
    'Debit card':'借记卡',
    'Cheque':'支票',
    'ACH':'自动清算中心',
    'Cross-border':'跨境',
    'Cash Withdrawal':'提款',
    'Cash Deposit':'现金存款'
}
df_laundering['Payment_type'] = df_laundering['Payment_type'].map(Payment_type_cn)
```

In [15]:
```
#计算支付类型TOP10
Payment_type_top10 = df_laundering.groupby('Payment_type').size().sort_values(ascending=False)
pie1 = (
    Pie()
    .add('',[list(z) for z in zip(Payment_type_top10.index.to_list(),Payment_type_top10.values.tolist())])
    .set_global_opts(
        title_opts=opts.TitleOpts(title='Top10 支付类型'),
        legend_opts=opts.LegendOpts(is_show=True,pos_top='95%')
    )
    .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}:{d}%"))
)
pie1.render_notebook()
```

Out[15]:



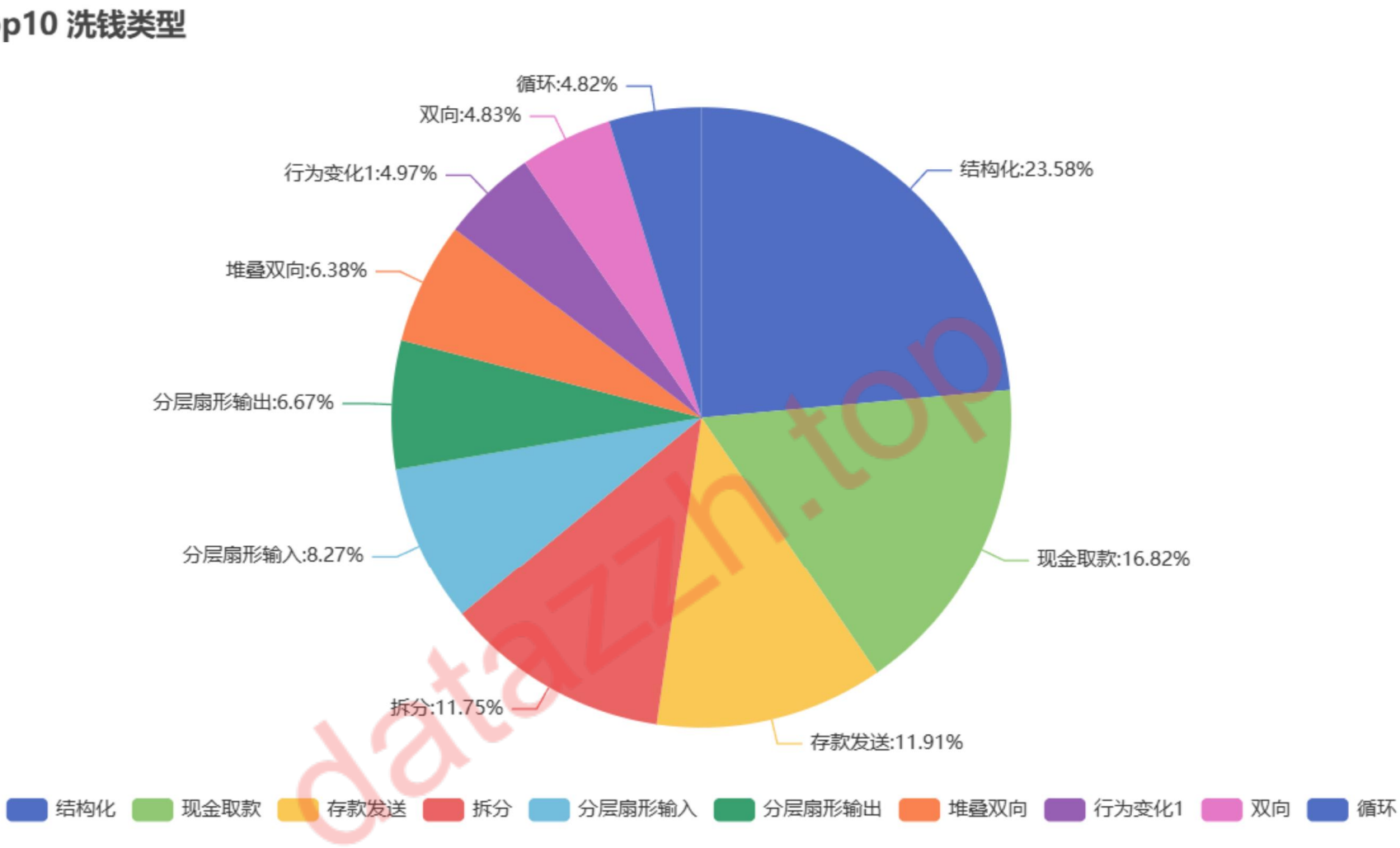
通过对TOP 10 支付类型饼状图可知，犯罪团伙涉案地区广，支付方式多样性

图四：Top10 洗钱类型

In [16]:
```
# 为更直观展示数据，对洗钱类型进行中文映射翻译
Laundering_type_cn = {
    'Structuring': '结构化',
    'Cash_Withdrawal': '现金取款',
    'Deposit-Send': '存款发送',
    'Smurfing': '拆分',
    'Layered_Fan_In': '分层扇形输入',
    'Layered_Fan_Out': '分层扇形输出',
    'Stacked Bipartite': '堆叠双向',
    'Behavioural_Change_1': '行为变化1',
    'Bipartite': '双向',
    'Cycle': '循环',
    'Fan_In': '扇形输入',
    'Gather-Scatter': '聚集-分散',
    'Behavioural_Change_2': '行为变化2',
    'Scatter-Gather': '分散-聚集',
    'Single_large': '单笔大额',
    'Fan_Out': '扇形输出',
    'Over-Invoicing': '超额发票'
}
df_laundering['Laundering_type']=df_laundering['Laundering_type'].map(Laundering_type_cn)
```

In [17]:
```
#计算洗钱类型TOP10
type_l = df_laundering.groupby('Laundering_type').size().sort_values(ascending=False).head(10)
pie2 = (
    Pie()
    .add('',[list(z) for z in zip(type_l.index.to_list(),type_l.values.tolist())])
    .set_global_opts(
        title_opts=opts.TitleOpts(title='Top10 洗钱类型'),
        legend_opts=opts.LegendOpts(is_show=True,pos_top='95%')
    )
    .set_series_opts(label_opts=opts.LabelOpts(formatter="{b}:{d}%"))
)
pie2.render_notebook()
```

Out[17]:



通过TOP10 洗钱类型饼状图可知，犯罪团伙的洗钱类型具有多样性、复杂性

Type *Markdown* and LaTeX: α^2

图五：每月洗钱量图

In [18]:
```
# 按月分组并计算每天的洗钱次数
grouped = df_laundering.groupby(pd.Grouper(key='DateTime', freq='M'))['Is_laundering'].count().reset_index(name='Laundering_count')
grouped = grouped.sort_values(by='DateTime')

# 绘制折线图
line = (
    Line()
    .add_xaxis(grouped['DateTime'].dt.date.tolist())  # 将时间转换为日期列表作为x轴
    .add_yaxis("洗钱次数", grouped['Laundering_count'].tolist(),
               is_symbol_show = True,
               label_opts=opts.LabelOpts(is_show=True)
              )  # 洗钱次数作为y轴
    .set_global_opts(
        title_opts=opts.TitleOpts(title="每月洗钱次数"),
    )
)

# 渲染图表
line.render_notebook()
```

Out[18]:

由图可见，平均每个月遭受到的洗钱事件高达898件

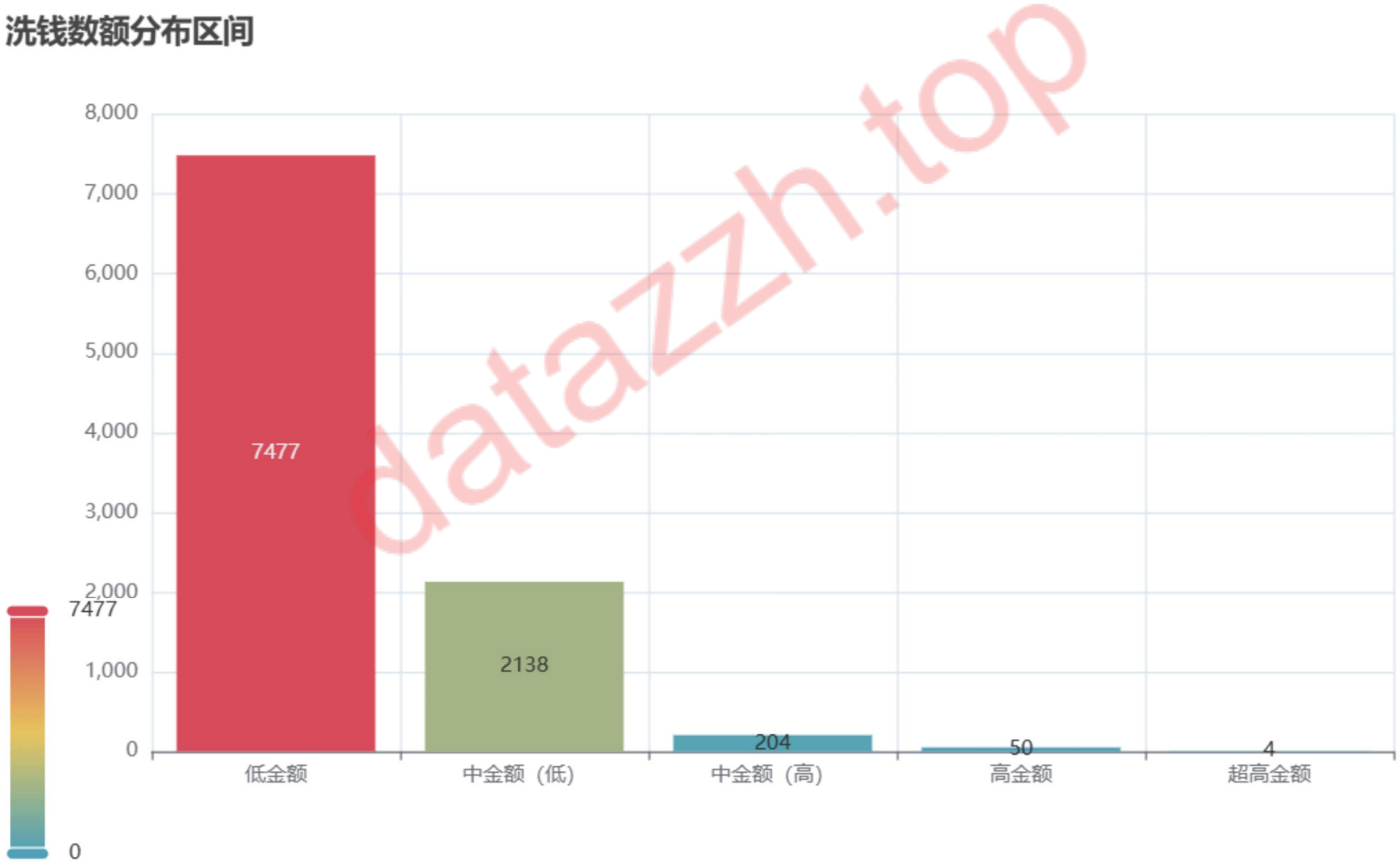
图六：洗钱份额分布

In [19]:
```
# 划分份额区间，便于特征列映射
def amount_range_set(amount):
    if amount < 10000:  # 0~10万
        return '低金额'
    elif 10000 <= amount < 100000:  # 1万~10万
        return '中金额（低）'
    elif 100000 <= amount < 1000000:  # 10万~100万
        return '中金额（高）'
    elif 1000000 <= amount < 10000000:  # 100万~1000万
        return '高金额'
    elif amount >= 10000000:  # 1000万以上
        return '超高金额'

df_laundering = df_laundering.copy()
df_laundering['Amount_Range'] = df_laundering['Amount'].apply(amount_range_set)
amount_range_counts = df_laundering.groupby('Amount_Range').size().reset_index(name='Count')
amount_range_counts.sort_values(by='Count',ascending=False,inplace=True)
```

In [20]:
```
bar5 = (
    Bar()
    .add_xaxis(amount_range_counts['Amount_Range'].to_list())
    .add_yaxis('',amount_range_counts['Count'].to_list())
    .set_global_opts(
        title_opts=opts.TitleOpts(title='洗钱数额分布区间'),
        visualmap_opts=opts.VisualMapOpts(max_=7477)
    )
)
bar5.render_notebook()
```

Out[20]:



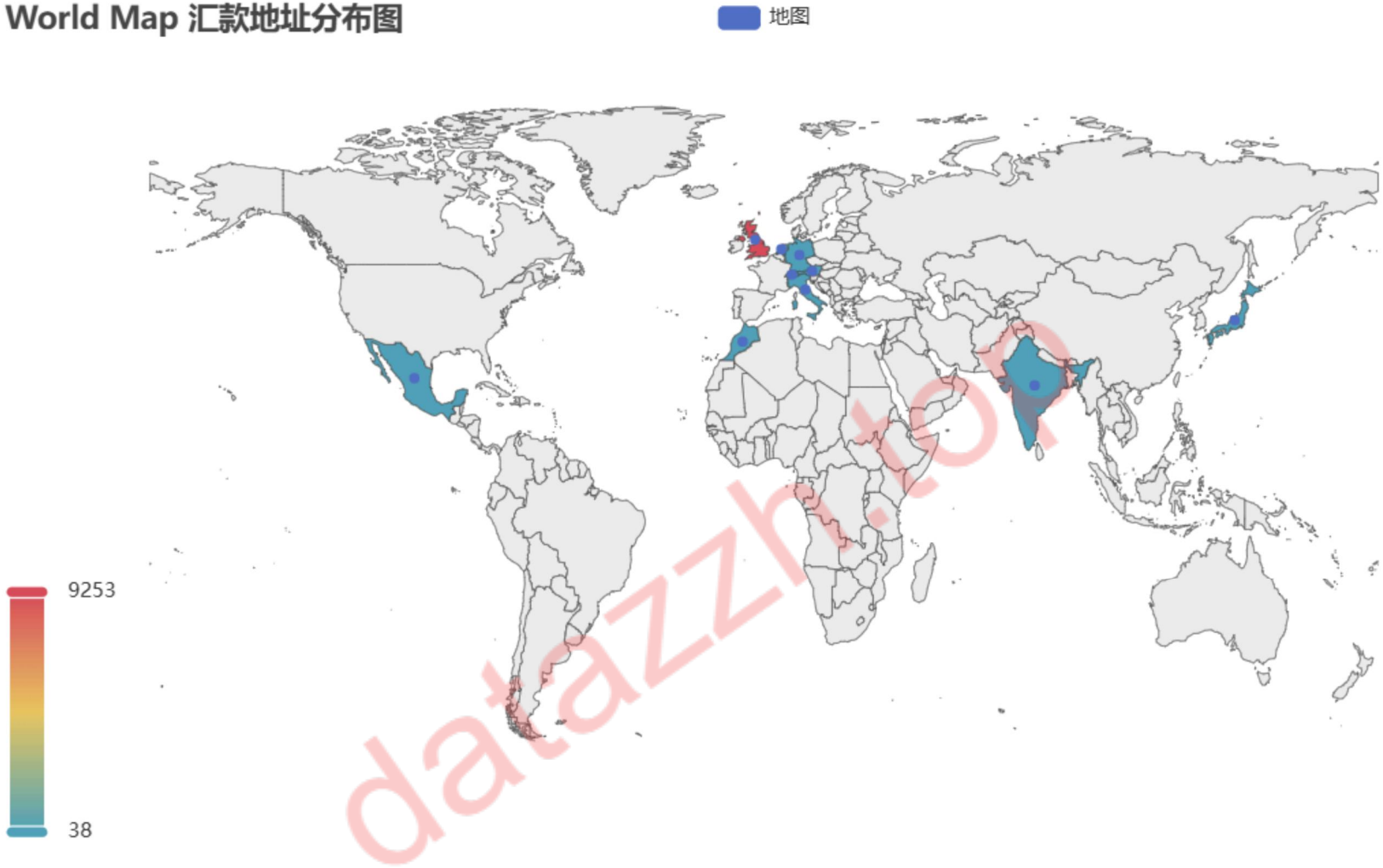
由此可见 绝大部分的洗钱案件份额大致在--10万，其中0--1万的份额居高。同时依图可见份额之间差距过大

图七、图八：汇\收款地区 World Map 分布图

In [21]:
```
# 补全国家地区名称，保证Map图形生成成功
def rename_guojia_name(guojia):
    if guojia == "UK":
        return "United Kingdom"
    elif guojia == "UAE":
        return "United Arab Emirates"
    else:
        return guojia
```

In [22]:
```
# 计算汇款前十地区
Sender_bank_location_top10 = df_laundering.groupby('Sender_bank_location').size().sort_values(ascending=False).head(10)
sbl_renamed = Sender_bank_location_top10.rename(index=rename_guojia_name)
sbl_dict = sbl_renamed.to_dict()
sbl_data_pair = list(sbl_dict.items())
# Map可视化
m1 = (
    Map()
    .add('地图',sbl_data_pair,maptype='world')
    .set_global_opts(
        title_opts=opts.TitleOpts(title="World Map 汇款地址分布图"),
        visualmap_opts=opts.VisualMapOpts(is_piecewise=False,
                                          min_=min(value for _, value in sbl_data_pair),  # 设置数据值范围的最小值
                                          max_=max(value for _, value in sbl_data_pair)  # 设置数据值范围的最大值
                                         )
    )
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))#取消国家名显示
)
m1.render_notebook()
```

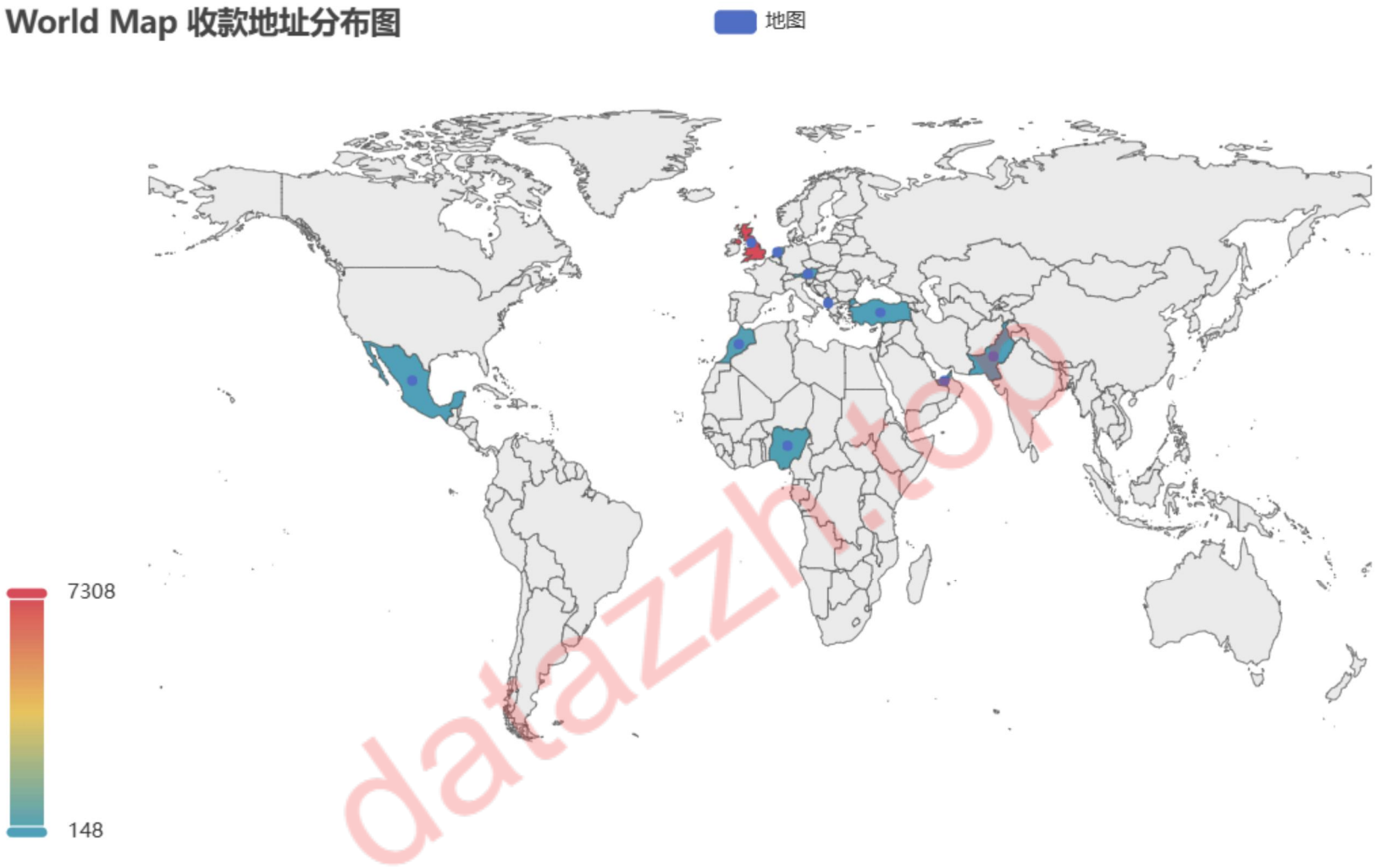
Out[22]:



In [23]:
```
# 计算收款前十地区
Receiver_bank_location_top10 = df_laundering.groupby('Receiver_bank_location').size().sort_values(ascending=False).head(10)
rbl_renamed = Receiver_bank_location_top10.rename(index=rename_guojia_name)
rbl_dict = rbl_renamed.to_dict()
rbl_data_pair = list(rbl_dict.items())
# Map可视化
m2 = (
    Map()
    .add('地图',rbl_data_pair,maptype='world')
    .set_global_opts(
        title_opts=opts.TitleOpts(title="World Map 收款地址分布图"),
        visualmap_opts=opts.VisualMapOpts(is_piecewise=False,
                                          min_=min(value for _, value in rbl_data_pair),  # 设置数据值范围的最小值
                                          max_=max(value for _, value in rbl_data_pair)  # 设置数据值范围的最大值
                                         )
    )
    .set_series_opts(label_opts=opts.LabelOpts(is_show=False))#取消国家名显示
)

m2.render_notebook()
```

Out[23]:



由此可见，英国本土发生的洗钱案件位居全球第一，同时是全球洗钱的体系核心

数据划分、数据建模

In [24]:
```
# 删除无用列
df.drop(columns=['DateTime','Sender_account','Receiver_account','Laundering_type','Payment_type'],inplace=True)
```

In [25]:
```
#对分类标签进行独热编码
col = ['Payment_currency','Received_currency','Sender_bank_location','Receiver_bank_location']
df = pd.get_dummies(df,columns=col)
```

In [26]:
```
# 划分数据集
y = df['Is_laundering']
x = df.drop(['Is_laundering'],axis=1)
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=42)
```

In [27]:
```
# 对数据进行特征缩放
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
model = LogisticRegression()
model.fit(x_train_scaled, y_train)
```

Out[27]:
```
▾ LogisticRegression

LogisticRegression()
```

模型评估与模型优化

In [28]:
```
# 打印模型准确率
y_predict = model.predict(x_test_scaled)
print('The accuracy of the Logistic Regression is:',accuracy_score(y_test,y_predict))
```

```
The accuracy of the Logistic Regression is: 0.906336438495533
```

In [29]:
```
# 展示模型评估报告
cr = classification_report(y_test,y_predict)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95     28490
           1       0.57      0.02      0.05      2963

    accuracy                           0.91     31453
   macro avg       0.74      0.51      0.50     31453
weighted avg       0.88      0.91      0.87     31453
```

In [*]:
```
#优化模型,选择模型最佳参数
parameters = {
    'penalty': ('l2',),
    'C': (0.01, 0.1, 1, 10)
}
grid_search = GridSearchCV(LogisticRegression(),parameters,verbose=0,scoring='accuracy',cv=10)
grid = grid_search.fit(x_train_scaled,y_train)
print('最佳效果:%0.3f' % grid_search.best_score_)
best_parameters= grid_search.best_estimator_.get_params()
print('最佳参数:\n',best_parameters)
```

通过网格搜索后，发现使用最佳参数调参的逻辑回归模型的准确率与前模型准确率接近

通过本次数据分析与建模实验，得出一下结论

1.全球面临着反洗钱的具体挑战任务

2.犯罪团伙在支付方式和洗钱类型具有多样性、复杂性的特点

3.洗钱的份额大多在0--10万区间内，0--1万的案件居多

4.每月洗钱案件平均高达898件，同时涉及洗钱案件最多的国家地区为英国，英国面临巨大的反洗钱问题

5.利用逻辑回归算法，对数据集进行划分建模，预测准确率结果高达0.91。通过评估报告可知模型对0类型的表现能力较好，对1类型表现能力较差。得出结论这是可能数据集不平衡、降噪不当导致的，同时行业内的反洗钱（AML）程序效率低下，法律和隐私限制，对交易监控数据的访问受到限制，可用数据缺乏真实的标签和多样性。故此模型只具有参考价值，因此，如何开发提高反洗钱的程序成为全球备受关注的一个热题