```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import f1_score

# 1. 数据导入和初步分析
data = pd.read_csv('/Users/kerwinlv/downloads/fraudulent.csv')
print("初步数据情况：")
print(data.info())
print(data.describe())
print("缺失值统计：")
print(data.isnull().sum())

# 2. 数据清洗与缺失值处理
# 删除缺失值超过50%的列
threshold = len(data) * 0.5
data = data.dropna(thresh=threshold, axis=1)

# 对其余缺失值进行填充（使用众数填充）
for col in data.columns:
    if data[col].isnull().sum() > 0:
        data[col].fillna(data[col].mode()[0], inplace=True)


# 3. 特征与标签提取
X = data.drop('y', axis=1)
y = data['y']
```

```
# 4. 数据集划分
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
test_size=0.2, random_state=1)

# 5. 特征缩放
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

# 6. 建立分类模型并进行训练
# (a) K-近邻模型
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred_val_knn = knn.predict(X_val)
f1_knn = f1_score(y_val, y_pred_val_knn)
print(f"KNN模型验证集 F1值: {f1_knn:.4f}")

# (b) 决策树模型
dt = DecisionTreeClassifier(random_state=1)
dt.fit(X_train, y_train)
y_pred_val_dt = dt.predict(X_val)
f1_dt = f1_score(y_val, y_pred_val_dt)
print(f"决策树模型验证集 F1值: {f1_dt:.4f}")

# (c) 逻辑回归模型
lr = LogisticRegression(max_iter=1000, random_state=1)
lr.fit(X_train, y_train)
y_pred_val_lr = lr.predict(X_val)
f1_lr = f1_score(y_val, y_pred_val_lr)
print(f"逻辑回归模型验证集 F1值: {f1_lr:.4f}")
```

```python
# (d) 支持向量机模型
svm = SVC(random_state=1)
svm.fit(X_train, y_train)
y_pred_val_svm = svm.predict(X_val)
f1_svm = f1_score(y_val, y_pred_val_svm)
print(f"支持向量机模型验证集 F1值: {f1_svm:.4f}")

# 7. 选择最佳模型并在测试集上评估
f1_scores = {'KNN': f1_knn, 'Decision Tree': f1_dt, 'Logistic
Regression': f1_lr, 'SVM': f1_svm}
best_model_name = max(f1_scores, key=f1_scores.get)
print(f"最佳模型: {best_model_name}, 验证集 F1值:
{f1_scores[best_model_name]:.4f}")

# 使用最佳模型在测试集上进行评估
if best_model_name == 'KNN':
    best_model = knn
elif best_model_name == 'Decision Tree':
    best_model = dt
elif best_model_name == 'Logistic Regression':
    best_model = lr
else:
    best_model = svm

y_pred_test = best_model.predict(X_test)
f1_test = f1_score(y_test, y_pred_test)
print(f"最佳模型 {best_model_name} 在测试集上的 F1值: {f1_test:.4f}")
```

```
KNN模型验证集 F1值: 0.8498
决策树模型验证集 F1值: 0.8759
逻辑回归模型验证集 F1值: 0.8561
支持向量机模型验证集 F1值: 0.8744
最佳模型: Decision Tree, 验证集 F1值: 0.8759
最佳模型 Decision Tree 在测试集上的 F1值: 0.8619

Process finished with exit code 0
```