```
import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import metrics
from sklearn.metrics import roc_curve
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from imblearn.over_sampling import SMOTE
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import f1_score, roc_auc_score, roc_curve, precision_recall_curve, auc, make_scorer, recall_score, accuracy_
from sklearn.model_selection import GridSearchCV
```

```
!cd fraudDetection/
!ls fraudDetection/
!pip install -U imbalanced-learn
!pip install pandas-profiling
!pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
```

```
cv_data.csv   imbalancedFraudDF.csv     test_data.csv   tr_server_data.csv
cv_label.csv  IpAddress_to_Country.csv  test_label.csv
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (0.10.1)
Collecting imbalanced-learn
  Downloading imbalanced_learn-0.12.2-py3-none-any.whl (257 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 258.0/258.0 kB 2.2 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.25.2)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (
ERROR: Operation cancelled by user
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pkg_resources/__init__.py", line 3108, in _dep_map
    return self.__dep_map
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pkg_resources/__init__.py", line 2901, in __getattr__
    raise AttributeError(attr)
AttributeError: _DistInfoDistribution__dep_map

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/pip/_internal/cli/base_command.py", line 169, in exc_logging_wrapper
    status = run_func(*args)
  File "/usr/local/lib/python3.10/dist-packages/pip/_internal/cli/req_command.py", line 242, in wrapper
    return func(self, options, args)
  File "/usr/local/lib/python3.10/dist-packages/pip/_internal/commands/install.py", line 441, in run
    conflicts = self._determine_conflicts(to_install)
  File "/usr/local/lib/python3.10/dist-packages/pip/_internal/commands/install.py", line 572, in _determine_conflicts
    return check_install_conflicts(to_install)
  File "/usr/local/lib/python3.10/dist-packages/pip/_internal/operations/check.py", line 101, in check_install_conflicts
    package_set, _ = create_package_set_from_installed()
  File "/usr/local/lib/python3.10/dist-packages/pip/_internal/operations/check.py", line 42, in create_package_set_from_
    dependencies = list(dist.iter_dependencies())
  File "/usr/local/lib/python3.10/dist-packages/pip/_internal/metadata/pkg_resources.py", line 216, in iter_dependencies
    return self._dist.requires(extras)
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pkg_resources/__init__.py", line 2821, in requires
    dm = self._dep_map
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pkg_resources/__init__.py", line 3110, in _dep_map
    self.__dep_map = self._compute_dependencies()
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pkg_resources/__init__.py", line 3120, in _compute_dependenc
    reqs.extend(parse_requirements(req))
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pkg_resources/__init__.py", line 3173, in __init__
    super(Requirement, self).__init__(requirement_string)
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/packaging/requirements.py", line 102, in __init__
    req = REQUIREMENT.parseString(requirement_string)
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 1131, in parse_string
    loc, tokens = self._parse(instring, 0)
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 817, in _parseNoCache
    loc, tokens = self.parseImpl(instring, pre_loc, doActions)
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 3886, in parseImpl
    loc, exprtokens = e._parse(instring, loc, doActions)
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 817, in _parseNoCache
    loc, tokens = self.parseImpl(instring, pre_loc, doActions)
  File "/usr/local/lib/python3.10/dist-packages/pip/_vendor/pyparsing/core.py", line 4114, in parseImpl
    return e._parse(
```

```
ipToCountry = pd.read_csv('fraudDetection/IpAddress_to_Country.csv')
fraud_data = pd.read_csv('fraudDetection/imbalancedFraudDF.csv')

fraud_data.head()
```

	user_id	signup_time	purchase_time	purchase_value	device_id	source	browser	sex	age	ip_address	class
0	22058	2015-02-24 22:55:49	2015-04-18 02:47:11	34	QVPSPJUOCKZAR	SEO	Chrome	M	39	7.327584e+08	0
1	333320	2015-06-07 20:39:50	2015-06-08 01:38:54	16	EOGFQPIZPYXFZ	Ads	Chrome	F	53	3.503114e+08	0
2	150084	2015-04-28 21:13:25	2015-05-04 13:54:50	44	ATGTXKYKUDUQN	SEO	Safari	M	41	3.840542e+09	0
[illegible]	[illegible]	2015-07-21	2015-09-09	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

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

```
class
0    136961
1      1415
Name: count, dtype: int64
```

```
# You can install pandas_profiling using the pip package manager by running:
# pip install pandas-profiling

import pandas_profiling

#Inline summary report without saving report as object
pandas_profiling.ProfileReport(fraud_data)

#simpler version without installing pandas_profiling
# fraud_data.describe().transpose()

# will give warnings on missing, correlation, constant value(0 variance), etc, see http://nbviewer.jupyter.org/github/JosPolflie
```

```
<ipython-input-6-85dec7efa125>:4: DeprecationWarning: `import pandas_profiling` is going to be deprecated by April 1st. Plea
  import pandas_profiling
```

Summarize dataset: 100% 36/36 [00:18<00:00, 3.49it/s, Completed]

Generate report structure: 100% 1/1 [00:08<00:00, 8.84s/it]

Render HTML: 100% 1/1 [00:02<00:00, 2.20s/it]

Overview

Dataset statistics

Number of variables	11
Number of observations	138376
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	11.6 MiB
Average record size in memory	88.0 B

Variable types

Numeric	4
DateTime	2
Text	1
Categorical	4

Alerts

`class` is highly imbalanced (91.8%)	Imbalance
`user_id` has unique values	Unique
`signup_time` has unique values	Unique

Reproduction

Analysis started	2024-04-07 23:25:42.194497

```
fraud_data.isna().sum()
```

```
user_id           0
signup_time       0
purchase_time     0
purchase_value    0
device_id         0
source            0
browser           0
sex               0
age               0
ip_address        0
class             0
dtype: int64
```

```
ipToCountry.head()
```

	lower_bound_ip_address	upper_bound_ip_address	country
0	16777216.0	16777471	Australia
1	16777472.0	16777727	China
2	16777728.0	16778239	China
3	16778240.0	16779263	Australia
4	16779264.0	16781311	China

```
start = time.time()

countries = []
for i in range(len(fraud_data)):
    ip_address = fraud_data.loc[i, 'ip_address']
    tmp = ipToCountry[(ipToCountry['lower_bound_ip_address'] <= ip_address) &
                    (ipToCountry['upper_bound_ip_address'] >= ip_address)]
    if len(tmp) == 1:
        countries.append(tmp['country'].values[0])
    else:
        countries.append('NA')

fraud_data['country'] = countries
runtime = time.time() - start

print("Lookup took", runtime, "seconds.")
```

```
Lookup took 181.20686292648315 seconds.
```

```
ip_address = fraud_data.loc[6, 'ip_address']
tmp = ipToCountry[(ipToCountry['lower_bound_ip_address'] <= ip_address) &
                    (ipToCountry['upper_bound_ip_address'] >= ip_address)]
print(tmp)
```

```
       lower_bound_ip_address  upper_bound_ip_address        country
28203            1.686110e+09              1694498815  United States
```

```
print(fraud_data.user_id.nunique())
print(len(fraud_data.index))
```

```
138376
138376
```

```
#Part3 Feature Engineering
```

```
fraud_data['interval_after_signup'] = (pd.to_datetime(fraud_data['purchase_time']) - pd.to_datetime(
        fraud_data['signup_time'])).dt.total_seconds()

fraud_data['signup_days_of_year'] = pd.DatetimeIndex(fraud_data['signup_time']).dayofyear

#bed time operation
fraud_data['signup_seconds_of_day'] = pd.DatetimeIndex(fraud_data['signup_time']).second + 60 * pd.DatetimeIndex(
    fraud_data['signup_time']).minute + 3600 * pd.DatetimeIndex(fraud_data['signup_time']).hour

fraud_data['purchase_days_of_year'] = pd.DatetimeIndex(fraud_data['purchase_time']).dayofyear
fraud_data['purchase_seconds_of_day'] = pd.DatetimeIndex(fraud_data['purchase_time']).second + 60 * pd.DatetimeIndex(
    fraud_data['purchase_time']).minute + 3600 * pd.DatetimeIndex(fraud_data['purchase_time']).hour

fraud_data = fraud_data.drop(['user_id','signup_time','purchase_time'], axis=1)
```

```
fraud_data.head()
```

	purchase_value	device_id	source	browser	sex	age	ip_address	class
0	34	QVPSPJUOCKZAR	SEO	Chrome	M	39	7.327584e+08	0
1	16	EOGFQPIZPYXFZ	Ads	Chrome	F	53	3.503114e+08	0
2	44	ATGTXKYKUDUQN	SEO	Safari	M	41	3.840542e+09	0
3	39	NAUITBZFJKHWW	Ads	Safari	M	45	4.155831e+08	0
4	42	ALEYXFXINSXLZ	Ads	Chrome	M	18	2.809315e+09	0

```
print(fraud_data.source.value_counts())
```

```
source
SEO       55766
Ads       54913
Direct    27697
Name: count, dtype: int64
```

```
#Part4 Feature Split
```

```
y = fraud_data['class']
X = fraud_data.drop(['class'], axis=1)

#split into train/test
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=0)
print("X_train.shape:", X_train.shape)
print("y_train.shape:", y_train.shape)
```

```
X_train.shape: (110700, 13)
y_train.shape: (110700,)
```

```
X_train['country'].value_counts(ascending=True)
```

```
country
Benin                 1
Yemen                 1
Fiji                  1
Monaco                1
Madagascar            1
                   ...
United Kingdom     3253
Japan              5251
China              8876
NA                16275
United States     42348
Name: count, Length: 177, dtype: int64
```

```
X_train.head()
```

	purchase_value	device_id	source	browser	sex	age	ip_address	country	interval_after_signup	signup_days_of_
29343	12	OULPAZAFRFPXP	Ads	Chrome	M	42	3.690922e+09	Korea Republic of	3499664.0	
12190	10	AIIWMFEYQQIEB	Ads	Opera	M	29	1.686759e+09	United States	6766039.0	
19388	34	VUVETBUPCIWJE	Direct	Chrome	M	53	4.138429e+09	NA	5870515.0	
89104	48	QCFULAJOYKFUU	Ads	Chrome	M	29	9.617337e+07	France	2145618.0	
82082	44	IHRWLMIJMEEEU	Ads	FireFox	M	24	1.936025e+09	China	7079059.0	

```
#Feature Engineer
```

```
X_train = pd.get_dummies(X_train, columns=['source', 'browser'])
X_train['sex'] = (X_train.sex == 'M').astype(int)

X_train_device_id_mapping = X_train.device_id.value_counts(dropna=False)
X_train['n_dev_shared'] = X_train.device_id.map(X_train_device_id_mapping)


X_train_ip_address_mapping = X_train.ip_address.value_counts(dropna=False)
X_train['n_ip_shared'] = X_train.ip_address.map(X_train_ip_address_mapping)

X_train_country_mapping = X_train.country.value_counts(dropna=False)
X_train['n_country_shared'] = X_train.country.map(X_train_country_mapping)

X_train = X_train.drop(['device_id','ip_address','country'], axis=1)


X_test = pd.get_dummies(X_test, columns=['source', 'browser'])
X_test['sex'] = (X_test.sex == 'M').astype(int)

X_test['n_dev_shared'] = X_test.device_id.map(X_test.device_id.value_counts(dropna=False))

X_test['n_ip_shared'] = X_test.ip_address.map(X_test.ip_address.value_counts(dropna=False))

X_test['n_country_shared'] = X_test.country.map(X_test.country.value_counts(dropna=False))

X_test = X_test.drop(['device_id','ip_address','country'], axis=1)
```

```
X_train.head()
```

	purchase_value	sex	age	interval_after_signup	signup_days_of_year	signup_seconds_of_day	purchase_days_of_year	pur
29343	12	1	42	3499664.0	183	67384	224	
12190	10	1	29	6766039.0	5	78146	84	
19388	34	1	53	5870515.0	197	81354	265	
89104	48	1	29	2145618.0	160	30920	185	
82082	44	1	24	7079059.0	111	71897	193	

```
#Compute the train minimum and maximum to be used for later scaling:
scaler = preprocessing.MinMaxScaler().fit(X_train[['n_dev_shared', 'n_ip_shared', 'n_country_shared']])
print(scaler.data_max_)

#transform the training data and use them for the model training
X_train[['n_dev_shared', 'n_ip_shared', 'n_country_shared']] = scaler.transform(X_train[['n_dev_shared', 'n_ip_shared', 'n_count

#before the prediction of the test data, apply the same scaler obtained from above, on X_test, not fitting a brandnew scaler on
X_test[['n_dev_shared', 'n_ip_shared', 'n_country_shared']] = scaler.transform(X_test[['n_dev_shared', 'n_ip_shared', 'n_country
```

```
[1. 1. 1.]
```

```
X_train.n_dev_shared.value_counts(dropna=False)
```

```
n_dev_shared
0.0    105427
0.2      4774
0.4       324
0.6       124
0.8        45
1.0         6
Name: count, dtype: int64
```

```
X_test.n_dev_shared.value_counts(dropna=False)
```

```
n_dev_shared
0.0    27330
0.2      334
0.4       12
Name: count, dtype: int64
```

```
#Model Training
```

```
logreg = LogisticRegression()

logreg.fit(X_train,y_train)

y_pred=logreg.predict(X_test)


cm = metrics.confusion_matrix(y_test, y_pred)
cmDF = pd.DataFrame(cm, columns=['pred_0', 'pred_1'], index=['true_0', 'true_1'])
print(cmDF)
```

```
        pred_0  pred_1
true_0   27389       0
true_1     287       0
```

```
classifier_RF = RandomForestClassifier(random_state=0)
classifier_RF.fit(X_train, y_train)
probs = classifier_RF.predict_proba(X_test)
predicted = classifier_RF.predict(X_test)

# generate evaluation metrics
print("%s: %r" % ("accuracy_score is: ", accuracy_score(y_test, predicted)))
print("%s: %r" % ("roc_auc_score is: ", roc_auc_score(y_test, probs[:, 1])))
print("%s: %r" % ("f1_score is: ", f1_score(y_test, predicted )))#string to int

print ("confusion_matrix is: ")
cm = confusion_matrix(y_test, predicted)
cmDF = pd.DataFrame(cm, columns=['pred_0', 'pred_1'], index=['true_0', 'true_1'])
print(cmDF)
print('recall =',float(cm[1,1])/(cm[1,0]+cm[1,1]))
print('precision =', float(cm[1,1])/(cm[1,1] + cm[0,1]))#1.0predicted = classifier_RF.predict(X_test)
```

```
accuracy_score is: : 0.9948692007515537
roc_auc_score is: : 0.7801672204169557
f1_score is: : 0.6712962962962962
confusion_matrix is: 
        pred_0  pred_1
true_0   27389       0
true_1     142     145
recall = 0.5052264808362369
precision = 1.0
```

```
smote = SMOTE(random_state=12)
x_train_sm, y_train_sm = smote.fit_resample(X_train, y_train)

unique, counts = np.unique(y_train_sm, return_counts=True)

print(np.asarray((unique, counts)).T)
```

```
[[     0 109572]
 [     1 109572]]
```

```
#RF on smoted training data
classifier_RF_sm = RandomForestClassifier(random_state=0)

classifier_RF_sm.fit(x_train_sm, y_train_sm)

# predict class labels for the test set
predicted_sm = classifier_RF_sm.predict(X_test)

# generate class probabilities
probs_sm = classifier_RF_sm.predict_proba(X_test)


# generate evaluation metrics
print("%s: %r" % ("accuracy_score_sm is: ", accuracy_score(y_test, predicted_sm)))
print("%s: %r" % ("roc_auc_score_sm is: ", roc_auc_score(y_test, probs_sm[:, 1])))
print("%s: %r" % ("f1_score_sm is: ", f1_score(y_test, predicted_sm )))#string to int

print ("confusion_matrix_sm is: ")
cm_sm = confusion_matrix(y_test, predicted_sm)
cmDF = pd.DataFrame(cm_sm, columns=['pred_0', 'pred_1'], index=['true_0', 'true_1'])
print(cmDF)
print('recall or sens_sm =',float(cm_sm[1,1])/(cm_sm[1,0]+cm_sm[1,1]))
print('precision_sm =', float(cm_sm[1,1])/(cm_sm[1,1] + cm_sm[0,1]))
```

```
accuracy_score_sm is: : 0.9948330683624801
roc_auc_score_sm is: : 0.7666438992331798
f1_score_sm is: : 0.6697459584295612
confusion_matrix_sm is:
        pred_0  pred_1
true_0   27388       1
true_1     142     145
recall or sens_sm = 0.5052264808362369
precision_sm = 0.9931506849315068
```

```
#Part 6: Parameter tuning by GridSearchCV


scorers = {
    'precision_score': make_scorer(precision_score),
    'recall_score': make_scorer(recall_score),
    'f1_score': make_scorer(f1_score, pos_label=1)

}


def grid_search_wrapper(model, parameters, refit_score='f1_score'):
    """
    fits a GridSearchCV classifier using refit_score for optimization(refit on the best model according to refit_score)
    for each combination of parameters, calculate all score in scorers, save them
    prints classifier performance metrics
    """

    grid_search = GridSearchCV(model, parameters, scoring=scorers, refit=refit_score,
                           cv=3, return_train_score=True)
    grid_search.fit(X_train, y_train)

    # make the predictions
    y_pred = grid_search.predict(X_test)
    y_prob = grid_search.predict_proba(X_test)[:, 1]

    print('Best params for {}'.format(refit_score))
    print(grid_search.best_params_)

    # confusion matrix on the test data.
    print('\nConfusion matrix of Random Forest optimized for {} on the test data:'.format(refit_score))
    cm = confusion_matrix(y_test, y_pred)
    cmDF = pd.DataFrame(cm, columns=['pred_0', 'pred_1'], index=['true_0', 'true_1'])
    print(cmDF)

    print("\t%s: %r" % ("roc_auc_score is: ", roc_auc_score(y_test, y_prob)))
    print("\t%s: %r" % ("f1_score is: ", f1_score(y_test, y_pred)))#string to int

    print('recall = ', float(cm[1,1]) / (cm[1,0] + cm[1,1]))
    print('precision = ', float(cm[1,1]) / (cm[1, 1] + cm[0,1]))

    return grid_search



# C: inverse of regularization strength, smaller values specify stronger regularization
LRGrid = {"C" : np.logspace(-2,2,5), "penalty":["l1","l2"]}# l1 lasso l2 ridge
#param_grid = {'C': [0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}
logRegModel = LogisticRegression(random_state=0)

grid_search_LR_f1 = grid_search_wrapper(logRegModel, LRGrid, refit_score='f1_score')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
```

```
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to conve
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, msg_start, len(result))
```

```python
parameters = {
#None: nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples
'max_depth': [None, 5, 15],
'n_estimators' :  [10,150],
'class_weight' : [{0: 1, 1: w} for w in [0.2, 1, 100]]
}

clf = RandomForestClassifier(random_state=0)
```

```python
grid_search_rf_f1 = grid_search_wrapper(clf, parameters, refit_score='f1_score')#no improvement on f1
```

```
Best params for f1_score
{'class_weight': {0: 1, 1: 0.2}, 'max_depth': None, 'n_estimators': 150}

Confusion matrix of Random Forest optimized for f1_score on the test data:
        pred_0  pred_1
true_0   27389       0
true_1     142     145
        roc_auc_score is: : 0.7781993788548851
        f1_score is: : 0.6712962962962962
recall =  0.5052264808362369
precision =  1.0
```

```python
best_rf_model_f1 = grid_search_rf_f1.best_estimator_
best_rf_model_f1
```

```
▾                         RandomForestClassifier
RandomForestClassifier(class_weight={0: 1, 1: 0.2}, n_estimators=150,
                       random_state=0)
```

```python
results_f1 = pd.DataFrame(grid_search_rf_f1.cv_results_)
results_sortf1 = results_f1.sort_values(by='mean_test_f1_score', ascending=False)
results_sortf1[['mean_test_precision_score', 'mean_test_recall_score', 'mean_test_f1_score', 'mean_train_precision_score', 'mear
```

	mean_test_precision_score	mean_test_recall_score	mean_test_f1_score	mean_train_precision_score	mean_train_recall_scor
9	1.0	0.527	0.69	1.0	0.52
1	1.0	0.527	0.69	1.0	1.00
13	1.0	0.527	0.69	1.0	1.00
3	1.0	0.527	0.69	1.0	0.52
5	1.0	0.527	0.69	1.0	0.56

```
pd.DataFrame(best_rf_model_f1.feature_importances_, index = X_train.columns, columns=['importance']).sort_values('importance', a
```

	importance
interval_after_signup	0.408875
purchase_days_of_year	0.132442
purchase_seconds_of_day	0.079075
signup_seconds_of_day	0.077661
signup_days_of_year	0.057319
n_ip_shared	0.052617
purchase_value	0.044106
age	0.038233
n_dev_shared	0.035686
n_country_shared	0.027432
sex	0.008170
source_Ads	0.006122
browser_Chrome	0.006042
source_SEO	0.005925
browser_Safari	0.004952
source_Direct	0.004812
browser_FireFox	0.004662
browser_IE	0.004603
browser_Opera	0.001265

```
grid_search_rf_recall = grid_search_wrapper(clf, parameters, refit_score='recall_score')
```

```
Best params for recall_score
{'class_weight': {0: 1, 1: 100}, 'max_depth': 5, 'n_estimators': 150}

Confusion matrix of Random Forest optimized for recall_score on the test data:
        pred_0  pred_1
true_0   27146     243
true_1     132     155
        roc_auc_score is: : 0.7904661234456265
        f1_score is: : 0.4525547445255475
recall =  0.5400696864111498
precision =  0.38944723618090454
```

```
best_RF_model_recall = grid_search_rf_recall.best_estimator_
best_RF_model_recall
```

```
▾                     RandomForestClassifier
RandomForestClassifier(class_weight={0: 1, 1: 100}, max_depth=5,
                       n_estimators=150, random_state=0)
```

```
# predict class labels for the test set
predictedBest_recall = best_RF_model_recall.predict(X_test)

# generate class probabilities
probsBest_recall = best_RF_model_recall.predict_proba(X_test)

results_recall = pd.DataFrame(grid_search_rf_recall.cv_results_)# recall score is different from above, as above is metric on te
results_sortrecall = results_recall.sort_values(by='mean_test_recall_score', ascending=False)
results_sortrecall[['mean_test_precision_score', 'mean_test_recall_score', 'mean_test_f1_score', 'mean_train_precision_score', '
#recall is worse than default rf?? no this is on test, but train recall is better
```

	mean_test_precision_score	mean_test_recall_score	mean_test_f1_score	mean_train_precision_score	mean_train_recall_scor
15	0.159	0.636	0.254	0.164	0.65
14	0.160	0.633	0.255	0.162	0.65
16	0.675	0.533	0.593	0.759	0.81
0	0.995	0.527	0.689	1.000	0.85

```
#for task 3, based on the above var importance
trainDF = pd.concat([X_train, y_train], axis=1)
pd.crosstab(trainDF["n_dev_shared"],trainDF["class"])
#the larger n_dev_shared, the higher rate of fraud
```

class	0	1
n_dev_shared		
0.0	104966	461
0.2	4403	371
0.4	152	172
0.6	37	87
0.8	13	32
1.0	1	5

```
fraud_data.groupby("class")[['interval_after_signup']].mean()
```

	interval_after_signup
class	
0	5.191179e+06
1	2.570226e+06

```
fraud_data.groupby("class")[['interval_after_signup']].median()#1
```

	interval_after_signup
class	
0	5194911.0