In [10]:
```
%conda install -c glemaitre imbalanced-learn
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with
flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.8.4
  latest version: 4.12.0

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: /home/ec2-user/anaconda3/envs/python3

  added / updated specs:
    - imbalanced-learn


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    imbalanced-learn-0.2.1     |           py36_0         117 KB  gl
emaitre
    pip-21.0.1                 |     pyhd8ed1ab_0         1.1 MB  co
nda-forge
    ------------------------------------------------------------
                                           Total:         1.2 MB

The following NEW packages will be INSTALLED:

  imbalanced-learn   glemaitre/linux-64::imbalanced-learn-0.2.1-py36
_0

The following packages will be DOWNGRADED:

  pip                                  21.3.1-pyhd8ed1ab_0 --> 21.0
.1-pyhd8ed1ab_0
```

```
Downloading and Extracting Packages
pip-21.0.1           | 1.1 MB    | ##################################
#### | 100%
imbalanced-learn-0.2 | 117 KB    | ##################################
#### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
# Preprocessing data
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.preprocessing import StandardScaler

# Pipeline
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

# Metrics
from sklearn.metrics import f1_score, recall_score

# Model
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

# Fine-tune
from sklearn.model_selection import RandomizedSearchCV

# Resample
#from imblearn.over_sampling import SMOTE
#from imblearn.under_sampling import RandomUnderSampler
#from imblearn.pipeline import Pipeline
```

In [6]:
```
df_train = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')
```

In [8]:
```
df_train.head
```

Out[8]:
```
<bound method NDFrame.head of              Time        V1        V2
V3        V4        V5        V6  \
0        51742.0  1.217770  0.077244  0.498321  0.520815 -0.574932 -
0.708602
1        61413.0  0.665294 -0.779358  1.680700  3.158373 -1.445562
0.721120
2        74745.0  1.272176  0.169925 -0.179971  1.062221  0.521140
0.614676
3       144191.0 -7.725336 -2.354526 -4.607707  0.164617 -6.985642
5.637628
4        77866.0 -0.517416  0.313686  1.537756 -1.558396  0.700028
1.288011
...          ...       ...       ...       ...       ...       ...
...
170878  137492.0  2.065608 -0.067614 -1.328328  0.369224 -0.083189 -
1.306155
170879   80311.0  1.125243  0.177591  0.662601  1.400977 -0.434960 -
0.407135
170880  152359.0 -0.364390  0.658625  0.944379 -0.546945  0.681491
0.490597
170881   53008.0 -2.309209 -0.878879  0.393167  0.093302 -0.317286 -
1.093650
170882   33535.0  1.378581 -0.432810  0.447347 -0.723483 -0.846215 -
0.623103

              V7        V8        V9  ...       V21       V22
V23  \
0      -0.159101  0.039746  0.046620  ... -0.226154 -0.773018  0.145
243
1      -0.733349  0.351817  0.947691  ...  0.046143 -0.017651 -0.209
463
2       0.017768  0.087789  0.419398  ... -0.199115 -0.329121 -0.228
425
3       5.313883  0.660609  0.064677  ... -1.073901  1.173600 -0.183
291
4       0.271898  0.507856  0.038740  ...  0.010926  0.338478  0.066
529
...          ...       ...       ...  ...       ...       ...
...
170878  0.184584 -0.331638  0.726241  ...  0.240669  0.828977  0.017
507
170879 -0.022928  0.014309  0.281087  ... -0.161778 -0.362048  0.089
908
170880  0.754740  0.109186 -0.550647  ... -0.050980 -0.163972 -0.409
029
170881  2.048833 -0.691987  0.683857  ... -0.333244  0.478814  1.093
621
170882 -0.550514 -0.094255 -0.857615  ... -0.019899 -0.119151  0.121
412
```

```
                V24       V25       V26       V27       V28   Amount  C
lass
0        0.276868  0.129511  0.083098 -0.043997  0.004983     0.99
0
1        0.385115  0.296200  0.082171  0.025492  0.071064   214.04
0
2       -1.313537  0.844641 -0.224776  0.037536  0.000581     3.76
0
3       -0.584831  0.008990  0.618218 -0.579530 -0.606947  1395.00
0
4       -0.989922 -0.638208  0.806935  0.075750 -0.181417     7.68
0
...           ...       ...       ...       ...       ...      ...
...
170878  0.000430  0.224140 -0.102555 -0.021962 -0.061156     1.00
0
170879  0.375493  0.388550 -0.487451  0.039770  0.026172    14.89
0
170880 -1.126093  0.748767  0.480345 -0.100499 -0.083629    44.95
0
170881  0.732311 -0.204624  0.224597 -0.412733 -0.332877   250.60
0
170882  0.075020  0.246502 -0.389617  0.023148  0.015991     4.95
0

[170883 rows x 31 columns]>
```

In [9]:
```
df_test.head
```

Out[9]:
```
<bound method NDFrame.head of             Time        V1        V2
V3        V4        V5        V6  \
0       36710.0  1.143676 -0.171978  1.237362  0.873063 -1.047431 -0
.279318
1      120618.0  1.907730 -0.036936 -1.956628  0.397331  0.366717 -0
.967145
2      132724.0 -0.790452  0.283243  1.027728 -0.407375  1.455679  5
.596155
3      134681.0  2.073675  0.089413 -1.709240  0.434341  0.310710 -0
.924093
4       44200.0 -3.022721  2.554556 -1.638564 -2.870878  1.322196  2
.948135
...         ...       ...       ...       ...       ...       ...
...
56957  133112.0  1.570653 -1.583754 -1.120449 -0.388292 -1.150111 -1
.038408
56958   32379.0 -0.759559  0.034247  2.286726 -0.499932 -0.099805  1
.513654
56959   82830.0 -3.494007  3.104977 -0.712754 -0.184329 -1.292975 -0
.865084
56960  138961.0 -0.799777  0.775712  1.823625 -0.609019  0.007770  0
```

```
.092002
56961  134875.0 -0.254663  0.581127  1.358820 -0.445842  0.157070 -0
.056924

             V7        V8        V9  ...       V21       V22       V
23  \
0     -0.550681  0.028027  0.821509  ... -0.071636 -0.022851  0.0057
27
1      0.268785 -0.249648  0.327864  ...  0.259187  0.740099 -0.1450
26
2     -2.187891 -0.908526  1.156964  ...  2.351712 -0.462717 -0.6417
73
3      0.116996 -0.181403  0.643899  ... -0.374555 -1.020429  0.3566
53
4     -0.616557  1.683751  1.621315  ... -0.452205 -0.715412  0.1010
42
...         ...       ...       ...  ...       ...       ...       .
..
56957 -0.044120 -0.321186 -0.244527  ... -0.615552 -1.980661  0.3492
90
56958 -0.214169  0.593662  1.238307  ... -0.053619  0.116634 -0.3155
27
56959 -0.905367  1.704704 -0.963684  ...  0.605534  0.406751  0.0410
22
56960  0.546546  0.242746 -0.045457  ... -0.228062 -0.613145 -0.2081
24
56961  0.280984  0.162744  0.454134  ... -0.118055 -0.334173  0.0182
98

            V24       V25       V26       V27       V28  Amount  Cla
ss
0      0.454436  0.271051  0.311897  0.017754  0.032307   28.75
0
1     -0.472334  0.253968 -0.104448 -0.017707 -0.030692   72.51
0
2      0.708122  1.614838 -0.242104  0.312321  0.165323   51.70
0
3      0.540546 -0.296108  0.173093 -0.069053 -0.032430    0.89
0
4      0.990957  0.361458  0.701487  0.256053 -0.252476    4.42
0
...         ...       ...       ...       ...       ...     ...    .
..
56957 -0.125742 -0.836879 -0.562662 -0.054156  0.014145  303.43
0
56958 -1.276704  0.177872  0.525804  0.087479  0.084239   43.50
0
56959  0.310343  0.129884 -0.682686 -2.054188 -0.206824    1.93
0
56960 -0.396119  0.366889 -0.438905  0.297092  0.123266   36.44
```

```
0
56961  0.598071 -0.603189 -0.606034  0.185155  0.183500    2.19
0

[56962 rows x 31 columns]>
```

In [10]:
```
X_train = df_train.drop(['Class'], axis=1)
y_train = df_train['Class']
X_test = df_test.drop(['Class'], axis=1)
y_test = df_test['Class']
```

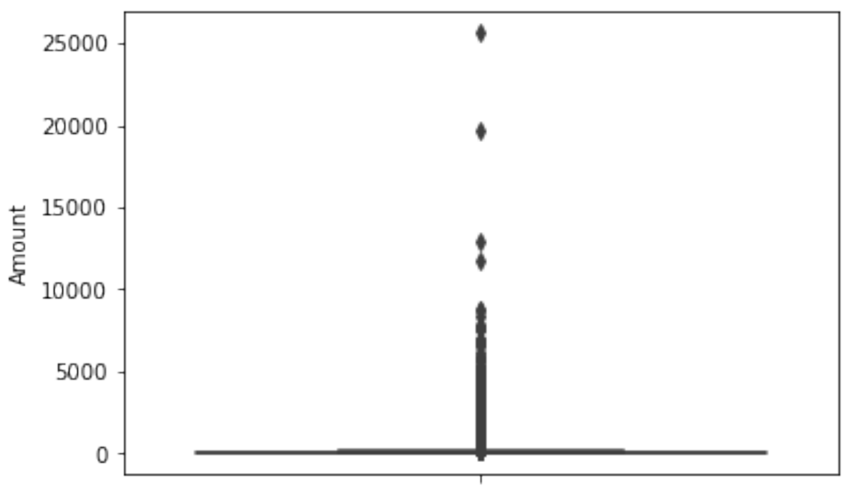
In [11]:
```
label_weight_perc = df_train['Class'].value_counts(normalize=True) * 1
00

label_weight_perc
```

Out[11]:
```
0    99.827367
1     0.172633
Name: Class, dtype: float64
```

In [12]:
```
#outliers
sns.boxplot(data= df_train, y='Amount')

plt.show()
```



In [13]:
```
feature_outlier = dict()

class Outlier:
    def __init__(self, q1, q3):
        self.q1 = q1
        self.q3 = q3
        self.iqr = q3 - q1

    def get_outlier_boundary(self):
        lower_fence = self.q1 - 1.5 * self.iqr
        upper_fence = self.q3 + 1.5 * self.iqr

        return lower_fence, upper_fence


def filter_outlier(df, cols=[]):
    if 'is_outlier' not in df.columns:
        df['is_outlier'] = (False) * len(df)

    for col in cols:
        if col in feature_outlier.keys():
            outlier = feature_outlier[col]
        else:
            q1 = df[col].quantile(0.25)
            q3 = df[col].quantile(0.75)

            outlier = Outlier(q1, q3)
            feature_outlier[col] = outlier

        lower_fence, upper_fence = outlier.get_outlier_boundary()

        outlier = (df[col] < lower_fence) | (df[col] > upper_fence)

        df['is_outlier'] = outlier | df['is_outlier']

    df = df[~df['is_outlier']]
    df = df.drop(['is_outlier'], axis=1)

    return df
```

In [14]:
```
train_df = filter_outlier(df_train, cols=['Amount'])
```

In [15]:
```
#feature scaling
std_feat = ['Amount', 'Time']
std_pipeline = Pipeline([
    ('std_scaler', StandardScaler())
])
```

In [16]:
```
#transformation
```

In [17]:
```
full_pipeline = ColumnTransformer([
    ('std_feat', std_pipeline, std_feat)
], remainder='passthrough')
```

In [18]:
```
df_val = pd.read_csv('val.csv')
```

In [19]:
```
df_val.head
```

Out[19]:
```
<bound method NDFrame.head of            Time        V1        V2
V3        V4        V5        V6  \
0      129298.0 -0.954393  0.405783  1.606533 -0.680623  0.733790  1
.610591
1       81657.0 -2.155718  1.805114  0.149724 -0.614140 -1.176974 -0
.892422
2       67843.0  1.338498 -1.279876  1.136343 -0.538429 -1.533192  1
.061259
3      124780.0 -0.716713  1.178418  0.646792  0.955424  0.803746 -0
.338454
4       34407.0 -0.351584  0.623436  1.285965  3.061167 -0.631591  1
.123301
...         ...       ...       ...       ...       ...       ...
...
56957   66083.0 -0.454076 -0.759274  0.471055 -2.945974  1.190941  3
.829154
56958  139370.0 -1.343480  2.194426  1.418332  4.356981 -0.013330  1
.315333
56959   51499.0  1.109835 -0.386229  0.500141  0.290532 -0.921449 -0
.657294
56960   48846.0 -1.532557 -1.710248  2.924359 -1.854211 -1.478623  0
.353205
56961   44922.0 -0.870968  0.522556  1.623563  0.890315  0.945442  0
.394801

             V7        V8        V9  ...       V21       V22       V
23  \
0      0.030525  0.747146  0.157591  ... -0.061662 -0.104440 -0.1601
79
1     -0.576635  0.973941 -0.367226  ...  0.255081  0.003716  0.1469
38
2     -1.810900  0.553728  0.834068  ...  0.094447  0.496196 -0.2285
42
3      0.696472 -0.317573 -0.236217  ... -0.435289 -0.990298 -0.3211
81
4     -1.090474 -2.461380 -1.641663  ... -1.261884 -0.125416 -0.1279
57
...         ...       ...       ...  ...       ...       ...       .
```

```
..
56957 -0.690131  0.933573 -2.348172  ... -0.405480 -0.968154 -0.0381
68
56958 -0.272056  1.082022 -2.277455  ... -0.328595 -1.055095  0.0264
34
56959 -0.237912  0.064436  0.550233  ... -0.184039 -0.587210  0.0344
98
56960 -0.371091 -0.174797 -0.712667  ... -0.542344 -0.326051 -0.1663
56
56961  1.076474 -0.667649 -0.194444  ...  0.051132  0.879997 -0.1119
66

            V24       V25       V26       V27       V28  Amount  Cla
ss
0     -1.717428  0.191840 -0.536290  0.374600  0.119729   34.00
0
1      0.464673 -0.323655  0.844538 -1.382840 -0.153149    7.68
0
2     -0.821732  0.535697  0.015714  0.066872  0.001772    9.20
0
3      0.646522  0.087046  0.695935 -0.245214 -0.063101    5.57
0
4     -0.018005  0.366680  0.070389  0.069952  0.272637  199.00
0
...         ...       ...       ...       ...       ...     ...    .
..
56957  0.961116  0.433040 -0.284959  0.062273  0.069019   76.93
0
56958  0.573707  0.029888  0.042230 -0.014299  0.020748   18.92
0
56959  0.556348  0.178829  0.881725 -0.095582 -0.000805   54.57
0
56960  0.536967  0.349679 -0.378818 -0.208043 -0.141084  160.00
0
56961 -0.277238 -0.257313 -0.292272 -0.733738 -0.563742   27.96
0

[56962 rows x 31 columns]>
```

In [20]:

```
X_val = df_test.drop(['Class'], axis=1)
y_val = df_test['Class']
```

In [21]:

```
X_train = train_df.drop(['Class'], axis=1)
y_train = train_df['Class']

X_train = full_pipeline.fit_transform(X_train)
```

In [22]:
```
X_val = df_val.drop(['Class'], axis=1)
y_val = df_val['Class']

X_val = full_pipeline.transform(X_val)
```

In [23]:
```
#model evaluation

model_eval = {
    'model': [],
    'recall': [],
    'f1_score': []
}

def add_model_eval(model, recall, f1_score):
    model_eval['model'].append(model)
    model_eval['recall'].append(f'{recall: .2f}')
    model_eval['f1_score'].append(f'{f1_score: .2f}')

def view_models_eval(sort=False):
    eval_df = pd.DataFrame(model_eval)

    if sort:
        eval_df = eval_df.sort_values(by=['recall', 'f1_score'], ascen
ding=[False, False])

    display(eval_df.style.hide_index())
```

Regression machine learning model

In [24]:
```
log_reg = LogisticRegression(random_state=42, verbose=1)
log_reg.fit(X_train, y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurr
ent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    3.2s finishe
d
```

Out[24]:
```
LogisticRegression(random_state=42, verbose=1)
```

In [25]:
```
y_pred = log_reg.predict(X_val)

add_model_eval('logistic regression', recall_score(y_val, y_pred), f1_
score(y_val, y_pred))
```

In [26]:
```
view_models_eval()
```

	model	recall	f1_score
	logistic regression	0.54	0.63

In [27]:
```
sgd_clf = SGDClassifier(random_state=42, verbose=1)
sgd_clf.fit(X_train, y_train)
```

```
-- Epoch 1
Norm: 24.58, NNZs: 30, Bias: -250.733088, T: 151848, Avg. loss: 0.35
4713
Total training time: 0.04 seconds.
-- Epoch 2
Norm: 23.67, NNZs: 30, Bias: -246.761162, T: 303696, Avg. loss: 0.10
3502
Total training time: 0.07 seconds.
-- Epoch 3
Norm: 21.33, NNZs: 30, Bias: -244.602658, T: 455544, Avg. loss: 0.09
6837
Total training time: 0.11 seconds.
-- Epoch 4
Norm: 21.76, NNZs: 30, Bias: -242.914858, T: 607392, Avg. loss: 0.09
7192
Total training time: 0.14 seconds.
-- Epoch 5
Norm: 21.74, NNZs: 30, Bias: -241.629791, T: 759240, Avg. loss: 0.09
4501
Total training time: 0.18 seconds.
-- Epoch 6
Norm: 22.21, NNZs: 30, Bias: -240.541613, T: 911088, Avg. loss: 0.09
3422
Total training time: 0.22 seconds.
-- Epoch 7
Norm: 21.98, NNZs: 30, Bias: -239.678363, T: 1062936, Avg. loss: 0.0
91269
Total training time: 0.26 seconds.
-- Epoch 8
Norm: 21.93, NNZs: 30, Bias: -238.910134, T: 1214784, Avg. loss: 0.0
91346
Total training time: 0.29 seconds.
-- Epoch 9
Norm: 21.97, NNZs: 30, Bias: -238.225774, T: 1366632, Avg. loss: 0.0
91628
Total training time: 0.33 seconds.
-- Epoch 10
Norm: 22.07, NNZs: 30, Bias: -237.610065, T: 1518480, Avg. loss: 0.0
91402
```

```
Total training time: 0.37 seconds.
-- Epoch 11
Norm: 21.96, NNZs: 30, Bias: -237.071801, T: 1670328, Avg. loss: 0.0
90899
Total training time: 0.40 seconds.
-- Epoch 12
Norm: 21.98, NNZs: 30, Bias: -236.567928, T: 1822176, Avg. loss: 0.0
90649
Total training time: 0.44 seconds.
Convergence after 12 epochs took 0.44 seconds
```

Out[27]:
```
SGDClassifier(random_state=42, verbose=1)
```

In [28]:
```
y_pred = sgd_clf.predict(X_val)

add_model_eval('sgd classifier', recall_score(y_val, y_pred), f1_score
(y_val, y_pred))
```

In [29]:
```
view_models_eval()
```

model	recall	f1_score
logistic regression	0.54	0.63
sgd classifier	0.47	0.59

Random Forest Tree

In [31]:
```
forest_clf = RandomForestClassifier(random_state=42, verbose=2, n_jobs
=4)
forest_clf.fit(X_train, y_train)
```

```
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurre
nt workers.
```

```
building tree 1 of 100
building tree 2 of 100
building tree 3 of 100
building tree 4 of 100
building tree 5 of 100
building tree 6 of 100
building tree 7 of 100
building tree 8 of 100
building tree 9 of 100
building tree 10 of 100
building tree 11 of 100
building tree 12 of 100
building tree 13 of 100
building tree 14 of 100
building tree 15 of 100
building tree 16 of 100
building tree 17 of 100
building tree 18 of 100
building tree 19 of 100
building tree 20 of 100
building tree 21 of 100
building tree 22 of 100
building tree 23 of 100
building tree 24 of 100
building tree 25 of 100
building tree 26 of 100
building tree 27 of 100
building tree 28 of 100
building tree 29 of 100
building tree 30 of 100
building tree 31 of 100
building tree 32 of 100
building tree 33 of 100
building tree 34 of 100
building tree 35 of 100
building tree 36 of 100

[Parallel(n_jobs=4)]: Done  33 tasks      | elapsed:   28.9s

building tree 37 of 100
building tree 38 of 100
building tree 39 of 100
building tree 40 of 100
building tree 41 of 100
building tree 42 of 100
building tree 43 of 100
building tree 44 of 100
building tree 45 of 100
building tree 46 of 100
building tree 47 of 100
```

```
building tree 48 of 100
building tree 49 of 100
building tree 50 of 100
building tree 51 of 100
building tree 52 of 100
building tree 53 of 100
building tree 54 of 100
building tree 55 of 100
building tree 56 of 100
building tree 57 of 100
building tree 58 of 100
building tree 59 of 100
building tree 60 of 100
building tree 61 of 100
building tree 62 of 100
building tree 63 of 100
building tree 64 of 100
building tree 65 of 100
building tree 66 of 100
building tree 67 of 100
building tree 68 of 100
building tree 69 of 100
building tree 70 of 100
building tree 71 of 100
building tree 72 of 100
building tree 73 of 100
building tree 74 of 100
building tree 75 of 100
building tree 76 of 100
building tree 77 of 100
building tree 78 of 100
building tree 79 of 100
building tree 80 of 100
building tree 81 of 100
building tree 82 of 100
building tree 83 of 100
building tree 84 of 100
building tree 85 of 100
building tree 86 of 100
building tree 87 of 100
building tree 88 of 100
building tree 89 of 100
building tree 90 of 100
building tree 91 of 100
building tree 92 of 100
building tree 93 of 100
building tree 94 of 100
building tree 95 of 100
building tree 96 of 100
building tree 97 of 100
```

```
building tree 98 of 100
building tree 99 of 100
building tree 100 of 100

[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:  1.4min finishe
d
```

Out[31]:
```
RandomForestClassifier(n_jobs=4, random_state=42, verbose=2)
```

In [32]:
```
y_pred = forest_clf.predict(X_val)

add_model_eval('random forest classifier', recall_score(y_val, y_pred)
, f1_score(y_val, y_pred))
```

```
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurre
nt workers.
[Parallel(n_jobs=4)]: Done  33 tasks      | elapsed:    0.1s
[Parallel(n_jobs=4)]: Done 100 out of 100 | elapsed:    0.4s finishe
d
```

In [33]:
```
view_models_eval()
```

model	recall	f1_score
logistic regression	0.54	0.63
sgd classifier	0.47	0.59
random forest classifier	0.77	0.85