**Motivation:**

In this guide, you’ll learn how to get started with Anomaly Detection using Python, including:

1. What is Anamoly?
2. What are the challenges of Traditional methods to do Anamoly?
3. Using Isolation Forest for Anamoly Detection.
4. How to explain the results?

**Problem and Requirements:**

In finance Such as Banking , Ecommerce some users perform illegal transactions in the database and those transactions leave records in database. How ever this database is mostly composed of non criminal transactions from the people who comply with the rules of the system they are part of.

**Goal:** To identify this Fraudulant Transactions by means of running our Database through Anamoly detection system.

**Good news:** Our database has all the records for idenitfy the crime.No need for data gathering.

**Bad news:** Fraudlent transactions of the data is buried in millions of records and you might not recognise fradualnt transaction in the records.

**Challenges:**

Need for automation mainly because the sheer size of the records make the manual validation of the data to identify the anomaly is impossible to check all records.

If the problem is that the human expersts dont have the time to go through the dataset can we try to automate their decision process?

**Rule Based approach** - If experts could complie a set of rules for instance transactions that exceed a certain amount are anomaly or transactions of certain type X at certain time Y are considered anolmaly etc. Any type of clearly defined rule to singleout a fradualant transaction. With this rule we can easily implement a system that can easily flag any anamolous transaction.that fullfill the rules of anamoly.

* How ever they are several problems with this approach. Coming up with a good set of rules require a lot of expertise and work rule needs to be validated and for particular dataset at hand.
* If the dataset format changes or a different dataset is used with different distribution the complete set of rules need to be reworked to reflect the new situation.

Finally and most importantly its impossible to predict to all shapes and forms that anomalies can assume.

**Solution to the above challenge:** To address this problem of unknown nature of fraudulent transactions we can make use of data we have in the database. We have to build a system which learns from the data what constitues an anomly.

We have to put the requirement that the model has to be unsupervised meaning that when we train this model we will not be able to tell that which are anamoly or not. we dont have any information on that so far.

**Machine Learning approach:** first step we train a unsupervised model on our data to learn about

1. First step we train a unsupervised model on our data to learn about distribution of data and to get a sense of what an anamoly is in our context.
2. In step two we use this model to predict the likelihood of a transaction being fraud for every transaction using what it has learned in the above step.

A machine learning model will never give a definitive answer of whether a transaction is anamoly their will be always an inheriant uncertanity in the answers it give us. If our model gives a continous value as anamoly score that refelcts the model certanity of the transaction likely to be anamoly we can choose ourself how wide we want to cast our net while looking for possible anamoly.

The output of the model will be a mapping of every transaction to certain a anamoly score. This score reflects how certain our model is of certain transaction as anamoly or not. If we want to consider some subset of transaction we have to set a threshold. the nature of anamoly will help us to consider how wider net we want to perform . Downside is we might not undestand why a transaction is anamoly .thats why we have explanability to the model to understand why model has flagged a certain transaction as anamoly.

**Dataset:**

!pip install shap

Requirement already satisfied: shap in c:\users\my\anaconda3\lib\site-packages (0.41.0)  
Requirement already satisfied: cloudpickle in c:\users\my\anaconda3\lib\site-packages (from shap) (2.0.0)  
Requirement already satisfied: scikit-learn in c:\users\my\anaconda3\lib\site-packages (from shap) (1.0.2)  
Requirement already satisfied: packaging>20.9 in c:\users\my\anaconda3\lib\site-packages (from shap) (21.3)  
Requirement already satisfied: tqdm>4.25.0 in c:\users\my\anaconda3\lib\site-packages (from shap) (4.64.1)  
Requirement already satisfied: pandas in c:\users\my\anaconda3\lib\site-packages (from shap) (1.4.4)  
Requirement already satisfied: scipy in c:\users\my\anaconda3\lib\site-packages (from shap) (1.9.1)  
Requirement already satisfied: numpy in c:\users\my\anaconda3\lib\site-packages (from shap) (1.21.5)  
Requirement already satisfied: numba in c:\users\my\anaconda3\lib\site-packages (from shap) (0.55.1)  
Requirement already satisfied: slicer==0.0.7 in c:\users\my\anaconda3\lib\site-packages (from shap) (0.0.7)  
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\my\anaconda3\lib\site-packages (from packaging>20.9->shap) (3.0.9)  
Requirement already satisfied: colorama in c:\users\my\anaconda3\lib\site-packages (from tqdm>4.25.0->shap) (0.4.5)  
Requirement already satisfied: setuptools in c:\users\my\anaconda3\lib\site-packages (from numba->shap) (63.4.1)  
Requirement already satisfied: llvmlite<0.39,>=0.38.0rc1 in c:\users\my\anaconda3\lib\site-packages (from numba->shap) (0.38.0)  
Requirement already satisfied: pytz>=2020.1 in c:\users\my\anaconda3\lib\site-packages (from pandas->shap) (2022.1)  
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\my\anaconda3\lib\site-packages (from pandas->shap) (2.8.2)  
Requirement already satisfied: joblib>=0.11 in c:\users\my\anaconda3\lib\site-packages (from scikit-learn->shap) (1.1.0)  
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\my\anaconda3\lib\site-packages (from scikit-learn->shap) (2.2.0)  
Requirement already satisfied: six>=1.5 in c:\users\my\anaconda3\lib\site-packages (from python-dateutil>=2.8.1->pandas->shap) (1.16.0)

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.ensemble import IsolationForest  
from sklearn.metrics import f1\_score, precision\_score, recall\_score, roc\_curve  
from sklearn.metrics import plot\_precision\_recall\_curve, precision\_recall\_curve, roc\_auc\_score  
  
from tqdm import tqdm  
import shap  
shap.initjs()  
plt.rcParams["figure.figsize"] = (12,8)

","

"],l=[3,"

","

"],f=[1,'"],p={"\*":[1,"?

","

"],area:[1,"

","

"],col:[2,"

","

"],legend:[1,"

","

"],param:[1,"

","

"],tr:[2,"

","

data = pd.read\_csv(r'C:\Users\MY\Desktop\Blogs\Fraud\_Detection\_Blog\PS\_20174392719\_1491204439457\_log.csv')

data.head()

|  | step | type | amount | nameOrig | oldbalanceOrg | newbalanceOrig | nameDest | oldbalanceDest | newbalanceDest | isFraud | isFlaggedFraud |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | PAYMENT | 9839.64 | C1231006815 | 170136.0 | 160296.36 | M1979787155 | 0.0 | 0.0 | 0 | 0 |
| 1 | 1 | PAYMENT | 1864.28 | C1666544295 | 21249.0 | 19384.72 | M2044282225 | 0.0 | 0.0 | 0 | 0 |
| 2 | 1 | TRANSFER | 181.00 | C1305486145 | 181.0 | 0.00 | C553264065 | 0.0 | 0.0 | 1 | 0 |
| 3 | 1 | CASH\_OUT | 181.00 | C840083671 | 181.0 | 0.00 | C38997010 | 21182.0 | 0.0 | 1 | 0 |
| 4 | 1 | PAYMENT | 11668.14 | C2048537720 | 41554.0 | 29885.86 | M1230701703 | 0.0 | 0.0 | 0 | 0 |

**About Data:** step - maps a unit of time in the real world. In this case 1 step is 1 hour of time. Total steps 744 (30 days simulation). type - CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.

amount - amount of the transaction in local currency.

nameOrig - customer who started the transaction

oldbalanceOrg - initial balance before the transaction

newbalanceOrig - new balance after the transaction

nameDest - customer who is the recipient of the transaction

oldbalanceDest - initial balance recipient before the transaction. Note that there is not information for customers that start with M (Merchants).

newbalanceDest - new balance recipient after the transaction. Note that there is not information for customers that start with M (Merchants).

isFraud - This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control or customers accounts and try to empty the funds by transferring to another account and then cashing out of the system.

isFlaggedFraud - The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction.

data['isFraud'].value\_counts()/len(data)

0 0.998709  
1 0.001291  
Name: isFraud, dtype: float64

**Feature Engineering:**

features = pd.DataFrame(index = data.index)

numerical\_columns = ['amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest', 'newbalanceDest']

features[numerical\_columns] = data[numerical\_columns]

features['changebalanceOrig'] = features['newbalanceOrig'] - features['oldbalanceOrg']  
features['changebalanceDest'] = features['newbalanceDest'] - features['oldbalanceDest']

features['hour'] = data['step'] % 24

type\_one\_hot = pd.get\_dummies(data['type'])  
features = pd.concat([features, type\_one\_hot], axis=1)

**Isolation Forest:**

**Model:**

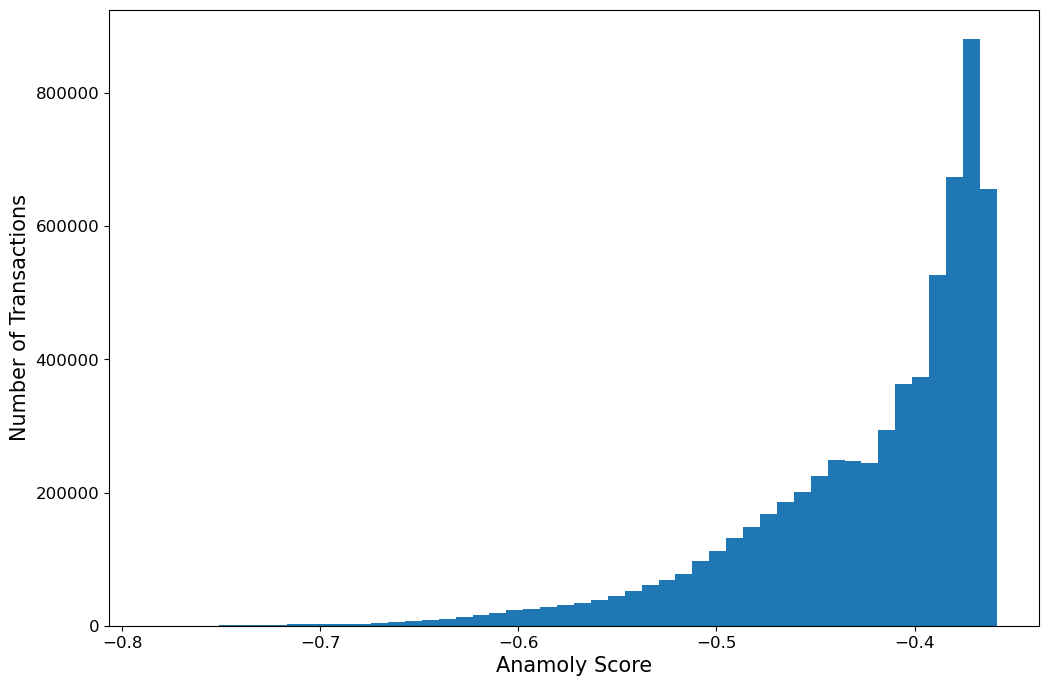
from sklearn.ensemble import IsolationForest  
  
forest = IsolationForest(random\_state = 0)  
forest.fit(features)

IsolationForest(random\_state=0)

scores = forest.score\_samples(features)

plt.hist(scores,bins =50)  
plt.ylabel('Number of Transactions',fontsize = 15)  
plt.xlabel('Anamoly Score',fontsize = 15)  
plt.xticks(fontsize = 12)  
plt.yticks(fontsize = 12)

(array([ 0., 200000., 400000., 600000., 800000., 1000000.]),  
 [Text(0, 0, ''),  
 Text(0, 0, ''),  
 Text(0, 0, ''),  
 Text(0, 0, ''),  
 Text(0, 0, ''),  
 Text(0, 0, '')])



**Top Outliers:**

top\_n\_outliers = 5  
top\_n\_outlier\_indices = np.argpartition(scores, top\_n\_outliers)[:top\_n\_outliers].tolist()  
top\_outlier\_entries = data.iloc[top\_n\_outlier\_indices, :]  
top\_outlier\_features = features.iloc[top\_n\_outlier\_indices, :]  
top\_outlier\_features

|  | amount | oldbalanceOrg | newbalanceOrig | oldbalanceDest | newbalanceDest | changebalanceOrig | changebalanceDest | hour | CASH\_IN | CASH\_OUT | DEBIT | PAYMENT | TRANSFER |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4157895 | 37387628.10 | 862621.88 | 0.0 | 37843025.59 | 7.523065e+07 | -862621.88 | 37387628.10 | 15 | 0 | 0 | 0 | 0 | 1 |
| 4146715 | 38874009.46 | 435622.96 | 0.0 | 73634878.73 | 1.125089e+08 | -435622.96 | 38874009.45 | 15 | 0 | 0 | 0 | 0 | 1 |
| 4153782 | 32572996.87 | 279467.23 | 0.0 | 84947676.90 | 1.175207e+08 | -279467.23 | 32572996.87 | 15 | 0 | 0 | 0 | 0 | 1 |
| 4142016 | 51141938.17 | 310058.79 | 0.0 | 87081824.45 | 1.382238e+08 | -310058.79 | 51141938.17 | 15 | 0 | 0 | 0 | 0 | 1 |
| 4145988 | 31866120.39 | 503019.94 | 0.0 | 63539675.65 | 9.540580e+07 | -503019.94 | 31866120.39 | 15 | 0 | 0 | 0 | 0 | 1 |

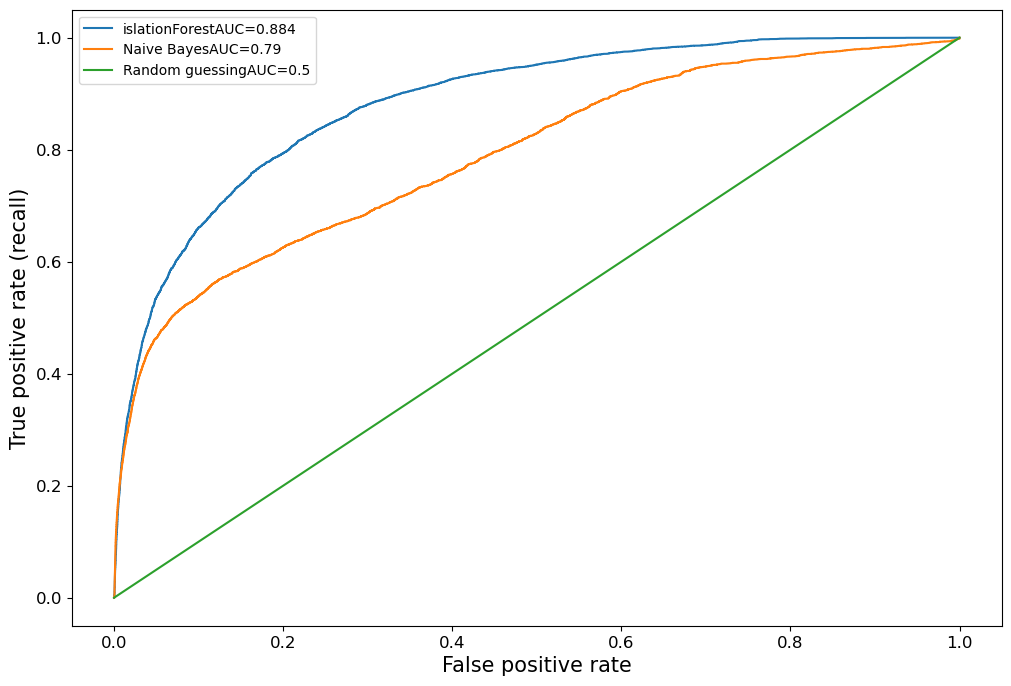
## Isolation forest anamoly score  
fpr\_forest,tpr\_forest,threshold\_forest = roc\_curve(data['isFraud'],-scores)  
auc\_score\_forest = roc\_auc\_score(data['isFraud'],-scores)  
  
## Naive Bayes Anamoly Score  
fpr\_naive,tpr\_naive,threshold\_naive = roc\_curve(data['isFraud'], data['amount'])  
auc\_score\_naive = roc\_auc\_score(data['isFraud'],data['amount'])

def plot\_roc\_curve(fpr,tpr,name,auc\_score):  
 plt.plot(fpr,tpr,label = name+'AUC={}'.format(round(auc\_score,3)))

**Results:**

plot\_roc\_curve(fpr\_forest,tpr\_forest,'islationForest',auc\_score\_forest)  
plot\_roc\_curve(fpr\_naive,tpr\_naive,'Naive Bayes',auc\_score\_naive)  
plot\_roc\_curve([0, 1], [0, 1], 'Random guessing', 0.5)  
plt.xlabel('False positive rate', fontsize=15)  
plt.ylabel('True positive rate (recall)', fontsize=15)  
plt.xticks(fontsize=12)  
plt.yticks(fontsize=12)  
plt.legend(prop={'size': 12})  
plt.legend()

<matplotlib.legend.Legend at 0x22633a36ac0>



**Explanability:**

import shap  
explainer = shap.TreeExplainer(forest)

random\_indices = np.random.choice(len(features),5000)  
shap\_values\_random = explainer.shap\_values(features.iloc[random\_indices,:])  
random\_features = features.iloc[random\_indices,:]

Explanation for local points

shap.initjs()

","

"],l=[3,"

","

"],f=[1,'"],p={"\*":[1,"?

","

"],area:[1,"

","

"],col:[2,"

","

"],legend:[1,"

","

"],param:[1,"

","

"],tr:[2,"

","

shap.initjs()  
shap.force\_plot(explainer.expected\_value, shap\_values\_random[1, :], random\_features.iloc[1, :])

","

"],l=[3,"

","

"],f=[1,'"],p={"\*":[1,"?

","

"],area:[1,"

","

"],col:[2,"

","

"],legend:[1,"

","

"],param:[1,"

","

"],tr:[2,"

","

<shap.plots.\_force.AdditiveForceVisualizer at 0x22633a81d60>

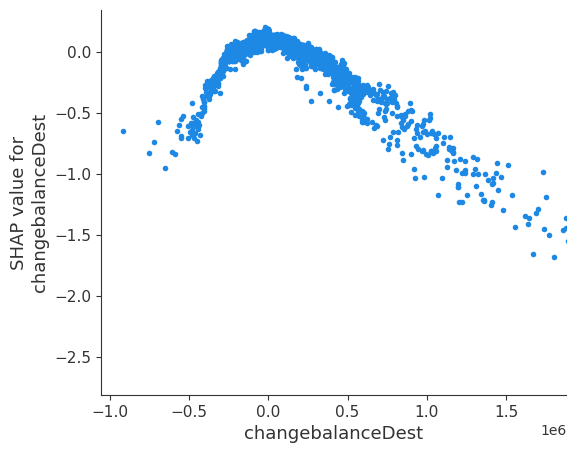
The above plot shows how the individual features of the data point contributed to shifting the model output from its expected to base value. These contributing values are called SHAP values. Blue values are contributing to lower the anamoly score and the red values for higher anamoly score.

For the above entry features such as amount(higher amount) is suggestive that the data is anamolous.

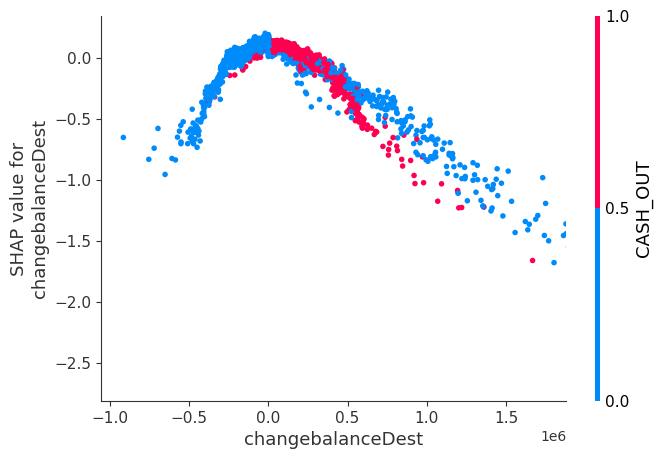
SHAP values are local explanations of different features for a given data point. To gain the global trends by aggregating the local explanations.

SHAP library has a library called dependence plot which shows us how mulitple datapoints of shap value depend on the feature .

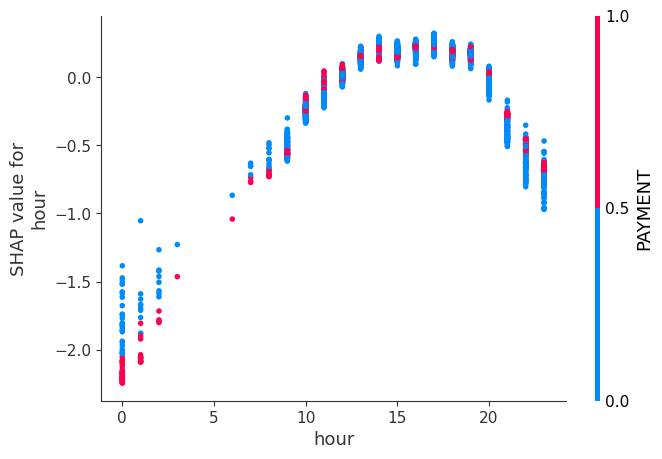
shap.dependence\_plot('changebalanceDest',shap\_values\_random,random\_features,interaction\_index= None,  
 xmax= 'percentile(99)')



shap.dependence\_plot(  
 'changebalanceDest',  
 shap\_values\_random,  
 random\_features,  
 interaction\_index='CASH\_OUT',  
 xmax='percentile(99)'  
)



shap.dependence\_plot(  
 'hour',  
 shap\_values\_random,  
 random\_features,  
 interaction\_index='PAYMENT',  
 xmax='percentile(99)'  
)



**Conclusion:**

The above guide demonstrates how to get started with Anomaly Detection using Python and the Isolation Forest algorithm. The guide explains the challenges of traditional methods for anomaly detection, such as rule-based approaches, and provides a solution to these challenges by using a machine learning approach. The guide uses the Isolation Forest algorithm to train a model on the data and predict the likelihood of a transaction being fraudulent based on the learned information. The guide also explains how to evaluate the results and provides code for visualizing the results using SHAP. The conclusion of the guide is that using a machine learning approach, specifically the Isolation Forest algorithm, can effectively identify fraudulent transactions in large datasets and provide a certain level of certainty in its predictions.