IsolationForest_0.05CV

June 29, 2020

[70]:
```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[71]:
```
PATH =   r"/home/swastik/workspace/repo/PyResearch/training/"
PATHTest =   r"/home/swastik/workspace/repo/PyResearch/testing/"

rawdata = pd.read_csv(PATHTest+"part-067.csv")
splitData = train_test_split(rawdata, test_size= 0.3)

train = splitData[0]
test = splitData[1]
```

[72]:
```
df.isAnomaly.value_counts()
```

[72]:
```
False    40231
True        21
Name: isAnomaly, dtype: int64
```

[73]:
```
from sklearn.ensemble import IsolationForest
rng = np.random.RandomState(42)

clf = IsolationForest(max_samples='auto', n_estimators=100,
                      random_state=rng,
                      max_features=1.0,
                      #behaviour="new",
                      contamination=0.01)
most_important_names= ['Heap usage activity : (d/dx (MXBean(java.lang:
 ↪type=Memory).HeapMemoryUsage.used))']
clf.fit(train[most_important_names])
y_pred_train = clf.predict(test[most_important_names])
display(y_pred_train)

unique, counts = np.unique(y_pred_train, return_counts=True)
```

```
predictions = dict(zip(unique, counts))

predictions
```

```
array([1, 1, 1, ..., 1, 1, 1])
```

[73]:
```
{-1: 125, 1: 11951}
```

[74]:
```
expected = np.where(test['isAnomaly'] == True, -1, 1)
expected
```

[74]:
```
array([1, 1, 1, …, 1, 1, 1])
```

[75]:
```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_pred_train,  expected)# Accuracy
```

[75]:
```
array([[    0,   125],
       [    7, 11944]])
```

[76]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_pred_train,  expected)# Recall
```

[76]:
```
0.9890692282212653
```

[77]:
```
from sklearn.metrics import recall_score
recall_score(y_pred_train,  expected, average=None)# Precision
```

[77]:
```
array([0.        , 0.99941427])
```

[78]:
```
from sklearn.metrics import precision_score
precision_score(y_pred_train,  expected, average=None)
```

[78]:
```
array([0.        , 0.98964289])
```

[79]:
```
from sklearn.metrics import mean_absolute_error

mean_absolute_error(y_pred_train,  expected)
```

[79]:
```
0.02186154355746936
```

[80]:
```
def kde_target(var_name, df):

    plt.figure(figsize = (12, 6))

    # Plot the distribution for target == 0 and target == 1
```

```
    sns.kdeplot(df.loc[df['isAnomaly'] == -1, var_name], label = 'isAnomaly ==␣
↪-1')
    sns.kdeplot(df.loc[df['isAnomaly'] == 1, var_name], label = 'isAnomaly ==␣
↪1')

    # label the plot
    plt.xlabel(var_name); plt.ylabel('Density'); plt.title('%s Distribution' %␣
↪var_name)
    plt.legend();
```

[81]:
```
dfx = test
dfx['isAnomaly'] = y_pred_train
dfx.isAnomaly.value_counts()
```

```
/home/swastik/.local/lib/python3.6/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[81]:
```
 1    11951
-1      125
Name: isAnomaly, dtype: int64
```

[ ]:
```

```