In [2]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input direct

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 5GB to the current directory (/kaggle/working/) that gets preserved as output when you cre
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/usaids-model-future-contraceptive-use/product.csv
/kaggle/input/usaids-model-future-contraceptive-use/data_dictionary.csv
/kaggle/input/usaids-model-future-contraceptive-use/annual_cases.csv
/kaggle/input/usaids-model-future-contraceptive-use/Table_of_Contents.xlsx
/kaggle/input/usaids-model-future-contraceptive-use/Train (2).csv
/kaggle/input/usaids-model-future-contraceptive-use/Recommended_Supplementary_Data_Sources.docx
/kaggle/input/usaids-model-future-contraceptive-use/monthly_cases.csv
/kaggle/input/usaids-model-future-contraceptive-use/SampleSubmission (2).csv
/kaggle/input/usaids-model-future-contraceptive-use/service_delivery_site_data.csv
/kaggle/input/usaids-model-future-contraceptive-use/contraceptive_case_data_annual.csv
/kaggle/input/contra-boost/Contra_service_delivery_site_data.csv
```

In [3]:
```
%matplotlib inline

pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
import seaborn as sns
import matplotlib.pyplot as plt
import time


#from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split,GridSearchCV
import matplotlib.pylab as plt
#from sklearn.tree import export_graphviz
#from sklearn import tree

from xgboost import XGBRegressor
from sklearn.model_selection import RandomizedSearchCV
```

In [4]:
```
train=pd.read_csv('../input/usaids-model-future-contraceptive-use/Train (2).csv')
sub=pd.read_csv('../input/usaids-model-future-contraceptive-use/SampleSubmission (2).csv')
sd=pd.read_csv('../input/contra-boost/Contra_service_delivery_site_data.csv')
```

In [5]:
```
train.columns
```

Out[5]:
```
Index(['year', 'month', 'region', 'district', 'site_code', 'product_code',
       'stock_initial', 'stock_received', 'stock_distributed',
       'stock_adjustment', 'stock_end', 'average_monthly_consumption',
       'stock_stockout_days', 'stock_ordered'],
      dtype='object')
```

In [6]:
```
train.drop(columns={'stock_initial', 'stock_received','stock_adjustment',
                    'stock_end', 'average_monthly_consumption',
                    'stock_stockout_days', 'stock_ordered'},inplace=True)
```

In [7]:
```
test=pd.DataFrame(sub.ID.str.split('X',3).tolist(), columns = ['year','month','site_code','product_code'])
for col in test.columns:
    test[col] = test[col].str.strip()
column=['year','month']
for col in column:
    test[col]=test[col].astype(int)
test=pd.merge(test,sd,on='site_code',how='left')
test.rename(columns={'site_region':'region','site_district':'district'},inplace=True)
test=test[['year', 'month', 'region','district','site_code', 'product_code']]
```

In [8]:
```
train.head(2).append(test.head(2))
```

Out[8]:

	year	month	region	district	site_code	product_code	stock_distributed
0	2019	1	INDENIE-DJUABLIN	ABENGOUROU	C4001	AS27134	21.0
1	2019	1	INDENIE-DJUABLIN	ABENGOUROU	C4001	AS27132	3.0
0	2019	7	INDENIE-DJUABLIN	ABENGOUROU	C4001	AS21126	NaN
1	2019	7	INDENIE-DJUABLIN	ABENGOUROU	C4001	AS27134	NaN

In [9]:
```
train['train_or_test']='train'
test['train_or_test']='test'
df=pd.concat([train,test])
```

In [10]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

for col in ['region','district','site_code','product_code']:
    df[col]=  df[col].astype('str')
    df[col]= le.fit_transform(df[col])
```

In [11]:
```
train=df.loc[df.train_or_test.isin(['train'])]
test=df.loc[df.train_or_test.isin(['test'])]
train.drop(columns={'train_or_test'},axis=1,inplace=True)
test.drop(columns={'train_or_test'},axis=1,inplace=True)
```

```
/opt/conda/lib/python3.7/site-packages/pandas/core/frame.py:3997: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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

In [12]:
```
train['product_code'] = train['product_code'].astype('category')
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  """Entry point for launching an IPython kernel.
```

In [13]:
```
test['product_code'] = test['product_code'].astype('category')
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  """Entry point for launching an IPython kernel.
```

In [14]:
```
train =pd.get_dummies(train, drop_first = True)
```

In [15]:
```
test =pd.get_dummies(test, drop_first = True)
```

In [16]:
```
train.columns
```

Out[16]:
```
Index(['year', 'month', 'region', 'district', 'site_code', 'stock_distributed',
       'product_code_1', 'product_code_2', 'product_code_3', 'product_code_4',
       'product_code_5', 'product_code_6', 'product_code_7', 'product_code_8',
       'product_code_9', 'product_code_10'],
      dtype='object')
```

In [17]:
```
test.columns
```

Out[17]:
```
Index(['year', 'month', 'region', 'district', 'site_code', 'stock_distributed',
       'product_code_1', 'product_code_2', 'product_code_3', 'product_code_4',
       'product_code_5', 'product_code_6', 'product_code_7', 'product_code_8',
       'product_code_9', 'product_code_10'],
      dtype='object')
```

In [18]:
```
len(test),len(sub)
```

Out[18]:
```
(3089, 3089)
```

In [19]:
```
train.head()
```

Out[19]:

	year	month	region	district	site_code	stock_distributed	product_code_1	product_code_2	product_code_3	product_code_4	product
0	2019	1	11	0	119	21.0	0	0	0	0	
1	2019	1	11	0	119	3.0	0	0	1	0	
2	2019	1	11	0	119	22.0	0	1	0	0	
3	2019	1	11	0	119	0.0	0	0	0	0	
4	2019	1	11	0	119	2.0	0	0	0	0	

In [20]:
```
train.corr()
```

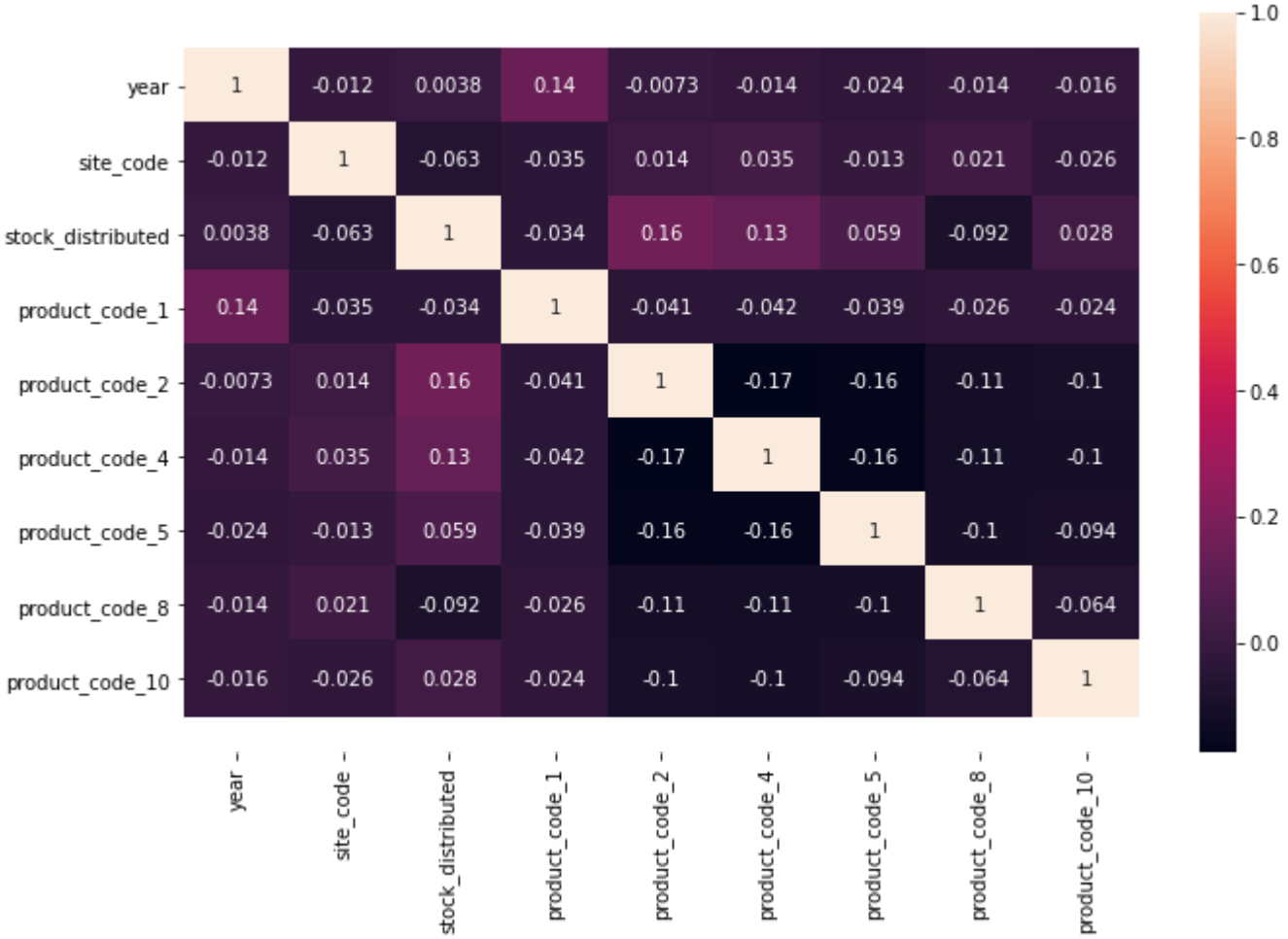
Out[20]:

	year	month	region	district	site_code	stock_distributed	product_code_1	product_code_2	product_code_
year	1.000000	-0.276131	-0.024058	-0.014014	-0.012142	0.003759	0.143073	-0.007298	-0.0093
month	-0.276131	1.000000	0.001509	0.002894	0.004769	0.002826	-0.036590	0.003318	0.0021
region	-0.024058	0.001509	1.000000	-0.010892	0.519487	-0.030961	-0.033480	0.004670	-0.0134
district	-0.014014	0.002894	-0.010892	1.000000	0.095592	-0.025359	0.001973	-0.021519	0.0052
site_code	-0.012142	0.004769	0.519487	0.095592	1.000000	-0.063014	-0.035154	0.013877	-0.0109
stock_distributed	0.003759	0.002826	-0.030961	-0.025359	-0.063014	1.000000	-0.033831	0.161034	-0.0884
product_code_1	0.143073	-0.036590	-0.033480	0.001973	-0.035154	-0.033831	1.000000	-0.041112	-0.0372
product_code_2	-0.007298	0.003318	0.004670	-0.021519	0.013877	0.161034	-0.041112	1.000000	-0.1562
product_code_3	-0.009338	0.002140	-0.013482	0.005243	-0.010936	-0.088402	-0.037259	-0.156297	1.0000
product_code_4	-0.014014	-0.001676	0.021134	-0.031569	0.034864	0.132680	-0.041611	-0.174550	-0.1581
product_code_5	-0.023701	0.002591	-0.019841	-0.020370	-0.012666	0.058682	-0.038552	-0.161721	-0.1465
product_code_6	0.006157	0.000328	0.014588	-0.001680	0.009215	-0.076340	-0.037321	-0.156558	-0.1418
product_code_7	0.043291	-0.002438	-0.006163	0.008002	0.010836	-0.072992	-0.035433	-0.148637	-0.1347
product_code_8	-0.013740	0.000145	0.050639	0.050441	0.021038	-0.091878	-0.026240	-0.110075	-0.0997
product_code_9	0.004799	0.017600	-0.007271	0.027587	-0.019056	-0.076946	-0.021075	-0.088405	-0.0801
product_code_10	-0.015862	-0.001555	-0.016770	-0.010028	-0.025928	0.028073	-0.023977	-0.100579	-0.0911

In [21]:
```
train.drop(columns = ['month', 'district', 'region', 'product_code_6', 'product_code_3', 'product_code_7', 'proc
test.drop(columns = ['month', 'district', 'region', 'product_code_6', 'product_code_3', 'product_code_7', 'produ
```

In [22]:
```
corr = train.corr()
fig, ax = plt.subplots()
fig.set_size_inches(11, 7)
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns,annot = True, ax = ax)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

Out[22]: (9.5, -0.5)



In [23]:
```
train['stock_distributed'] = np.where(train['stock_distributed'] == 0, train['stock_distributed'].median(), trai
```

In [24]:
```
X=train.drop(columns={'stock_distributed'})
y=train.loc[:,['stock_distributed']]
del test['stock_distributed']


train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, random_state=100)
```

#Define the model my_model = XGBRegressor()

booster = ['gbtree', 'gblinear'] base_score = [0.25, 0.5, 0.75, 1]

##Hyper Parameter Optimization n_estimators = [100, 500, 900, 1100, 1500] max_depth = [2, 3, 5, 10, 15] booster = ['gbtree', 'gblinear'] learning_rate = [0.05, 0.1, 0.15, 0.20] min_child_weight = [1, 2, 3, 4]

#Define the grid of hyperparameters to search hyperparameter_grid = { 'n_estimators' : n_estimators, 'max_depth': max_depth, 'learning_rate': learning_rate, 'min_child_weight': min_child_weight, 'booster': booster, 'base_score': base_score }

#Set up the random search with 4-fold cross validation random_cv = RandomizedSearchCV(estimator=my_model, param_distributions = hyperparameter_grid, cv = 5, n_iter = 50, scoring = 'neg_mean_absolute_error', n_jobs = 4, verbose = 5, return_train_score = True, random_state = 42)

#fit the model random_cv.fit(train_X, train_y)

random_cv.best_estimator_

In [25]:
```
#old parameters
my_model = XGBRegressor(base_score=0.25, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.1, max_delta_step=0, max_depth=10,
             min_child_weight=3,  missing=None, monotone_constraints='()',
             n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

In [26]:
```
my_model.fit(train_X, train_y)
pred = my_model.predict(test)
```

In [27]:
```
sub['prediction']=np.abs(pred)
```

In [28]:
```
sub['prediction'] = sub['prediction'].round()
```

In [29]:
```
sub.to_csv('XGBoost12.csv',index=False)
```

import keras from keras.models import Sequential from keras.layers import Dense from keras.layers import LeakyReLU,PReLU,ELU from keras.layers import Dropout

from keras import backend as K def root_mean_squared_error(y_true, y_pred): return K.sqrt(K.mean(K.square(y_pred - y_true)))

Initialising the ANN

classifier = Sequential()

Adding the input layer and the first hidden layer

classifier.add(Dense(10, kernel_initializer = 'he_uniform',activation='relu',input_dim = 6))

Adding the second hidden layer

classifier.add(Dense(5, kernel_initializer = 'he_uniform',activation='relu'))

Adding the third hidden layer

classifier.add(Dense(10, kernel_initializer = 'he_uniform',activation='relu'))

Adding the output layer

classifier.add(Dense(1, kernel_initializer = 'he_uniform'))

Compiling the ANN

classifier.compile(loss=root_mean_squared_error, optimizer='Adamax')

Fitting the ANN to the Training set

model_history=classifier.fit(train_X.values, train_y.values,validation_split=0.20, batch_size = 10, epochs = 1000)

ann_pred = classifier.predict(test)

sub['prediction']=np.abs(ann_pred)

sub['prediction']=sub['prediction'].round()

sub.to_csv('ANN.csv',index=False)