breast_cancer

November 9, 2020

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

[2]:
```
data = pd.read_csv("breast_cancer_data.csv")
```

[3]:
```
data.head()
```

[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	\
0	0.11840	0.27760	0.3001	0.14710	
1	0.08474	0.07864	0.0869	0.07017	
2	0.10960	0.15990	0.1974	0.12790	
3	0.14250	0.28390	0.2414	0.10520	
4	0.10030	0.13280	0.1980	0.10430	

	…	texture_worst	perimeter_worst	area_worst	smoothness_worst	\
0	…	17.33	184.60	2019.0	0.1622	
1	…	23.41	158.80	1956.0	0.1238	
2	…	25.53	152.50	1709.0	0.1444	
3	…	26.50	98.87	567.7	0.2098	
4	…	16.67	152.20	1575.0	0.1374	

	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	\
0	0.6656	0.7119	0.2654	0.4601	
1	0.1866	0.2416	0.1860	0.2750	
2	0.4245	0.4504	0.2430	0.3613	
3	0.8663	0.6869	0.2575	0.6638	
4	0.2050	0.4000	0.1625	0.2364	

```
   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

[4]:
```
data.columns
```

[4]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

[6]:
```
# We do not need the patient id for prediction
data = data.drop(['id'], axis = 1)
```

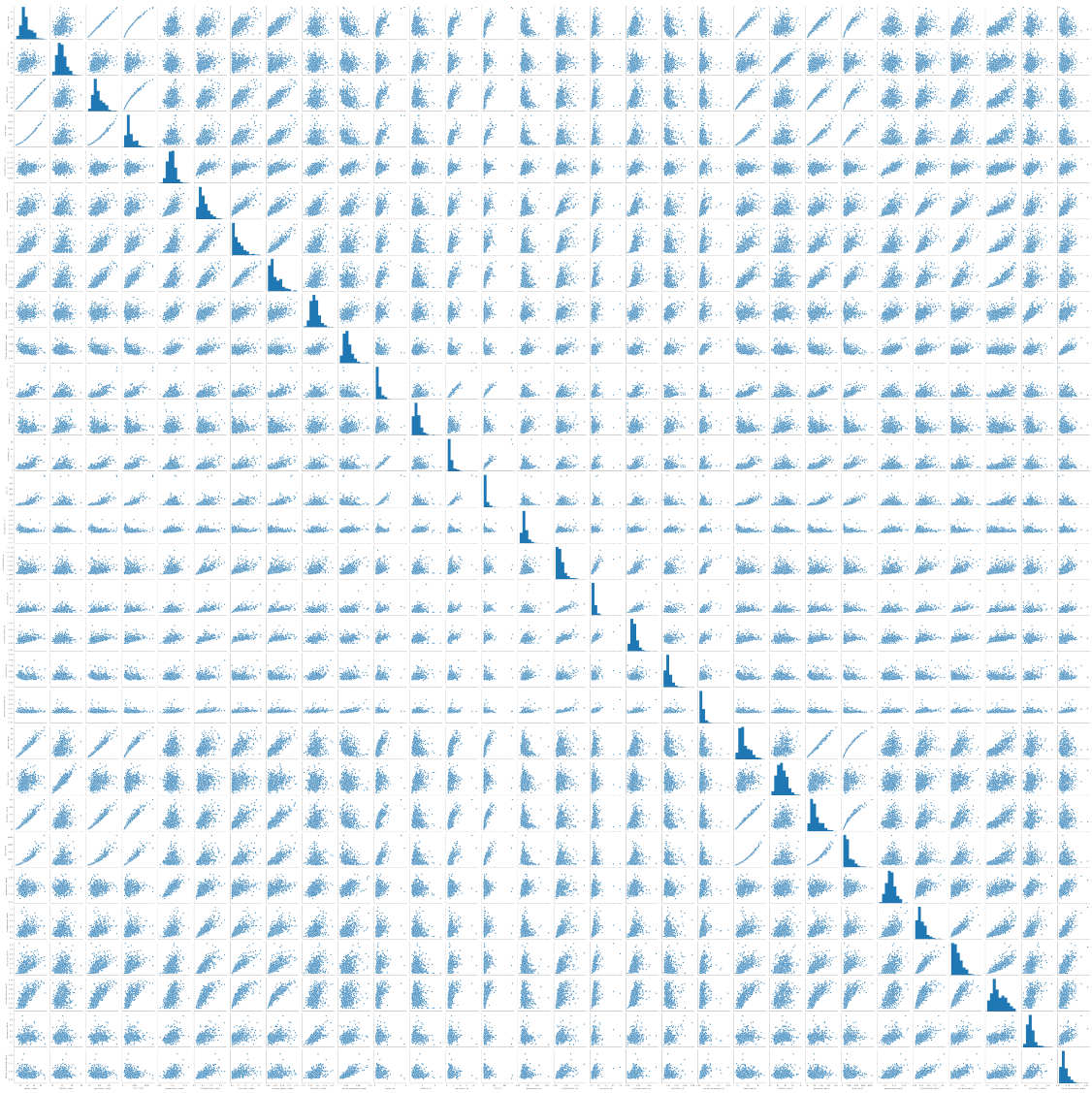
[7]:
```
# Summary numbers of the features
data.describe()
```

[7]:
```
       radius_mean  texture_mean  perimeter_mean    area_mean  \
count   569.000000    569.000000      569.000000   569.000000
mean     14.127292     19.289649       91.969033   654.889104
std       3.524049      4.301036       24.298981   351.914129
min       6.981000      9.710000       43.790000   143.500000
25%      11.700000     16.170000       75.170000   420.300000
50%      13.370000     18.840000       86.240000   551.100000
75%      15.780000     21.800000      104.100000   782.700000
max      28.110000     39.280000      188.500000  2501.000000

       smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
count       569.000000        569.000000      569.000000           569.000000
mean          0.096360          0.104341        0.088799             0.048919
std           0.014064          0.052813        0.079720             0.038803
min           0.052630          0.019380        0.000000             0.000000
25%           0.086370          0.064920        0.029560             0.020310
50%           0.095870          0.092630        0.061540             0.033500
75%           0.105300          0.130400        0.130700             0.074000
max           0.163400          0.345400        0.426800             0.201200
```

```
       symmetry_mean  fractal_dimension_mean  …  texture_worst  \
count     569.000000              569.000000  …     569.000000
mean        0.181162                0.062798  …      25.677223
std         0.027414                0.007060  …       6.146258
min         0.106000                0.049960  …      12.020000
25%         0.161900                0.057700  …      21.080000
50%         0.179200                0.061540  …      25.410000
75%         0.195700                0.066120  …      29.720000
max         0.304000                0.097440  …      49.540000

       perimeter_worst   area_worst  smoothness_worst  compactness_worst  \
count       569.000000   569.000000        569.000000         569.000000
mean        107.261213   880.583128          0.132369           0.254265
std          33.602542   569.356993          0.022832           0.157336
min          50.410000   185.200000          0.071170           0.027290
25%          84.110000   515.300000          0.116600           0.147200
50%          97.660000   686.500000          0.131300           0.211900
75%         125.400000  1084.000000          0.146000           0.339100
max         251.200000  4254.000000          0.222600           1.058000

       concavity_worst  concave points_worst  symmetry_worst  \
count       569.000000            569.000000      569.000000
mean          0.272188              0.114606        0.290076
std           0.208624              0.065732        0.061867
min           0.000000              0.000000        0.156500
25%           0.114500              0.064930        0.250400
50%           0.226700              0.099930        0.282200
75%           0.382900              0.161400        0.317900
max           1.252000              0.291000        0.663800

       fractal_dimension_worst  Unnamed: 32
count               569.000000          0.0
mean                  0.083946          NaN
std                   0.018061          NaN
min                   0.055040          NaN
25%                   0.071460          NaN
50%                   0.080040          NaN
75%                   0.092080          NaN
max                   0.207500          NaN

[8 rows x 31 columns]
```

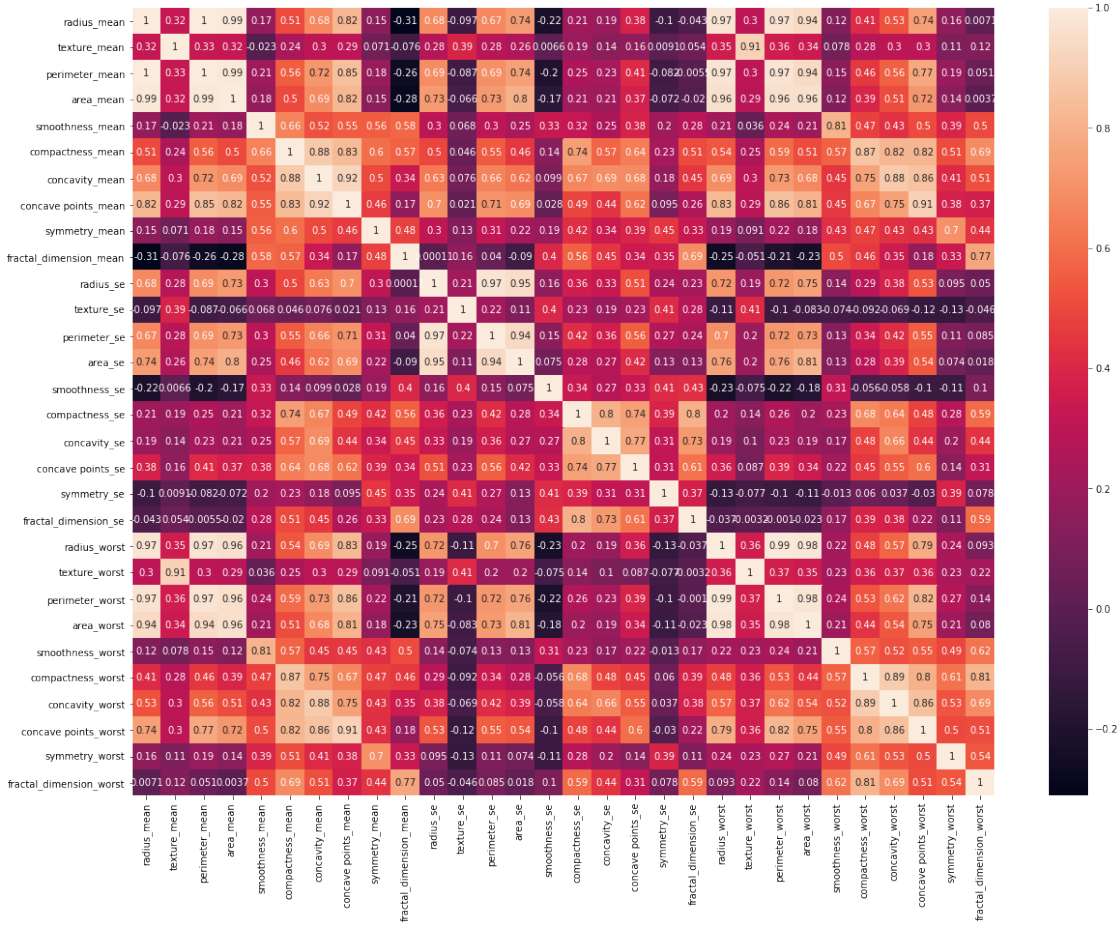
[12]:
```
# Creating a pairplot of the features to see how they are related to each other
# sns.pairplot(data.iloc[:,1:32])
```

[12]:
```
<seaborn.axisgrid.PairGrid at 0x7f122333e2b0>
```



[17]:
```
# Correlation matrix of features
corr_mat = data.iloc[:,1:32].corr()
plt.figure(figsize=(20,15))
sns.heatmap(corr_mat, annot = True)
```

[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7f11fe2bc9e8>



[38]:
```
import boto3, re, sys, math, json, os, sagemaker, urllib.request
from sagemaker import get_execution_role
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer
```

[62]:
```
# Defining IAM Role and setting up S3 bucket
role = get_execution_role()
prefix = 'sagemaker/breastcancer'
bucket_name = 'a20449643-csp554project' # Pre-created the bucket
my_region = boto3.session.Session().region_name
print(my_region)
```

```
us-east-1
```

[42]:
```
# Train-Test split
from sklearn.model_selection import train_test_split
```

```
X = data.iloc[:,1:32]
y = data['diagnosis']
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.20,␣
↪random_state=123)
```

[56]:
```
train_data = pd.concat([y_train, x_train], axis = 1)

test_data = pd.concat([y_test, x_test], axis = 1)
```

[57]:
```
# AWS Sagemaker requires data to be in csv format without headers
train_data.to_csv('train.csv', index = False, header = False)
```

[63]:
```
# Uploading training data to S3 bucket
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix,␣
↪'train/train.csv')).upload_file('train.csv')

# Setting up training data input path for sagemaker
s3_input_train = sagemaker.s3_input(s3_data='s3://{}/{}/train'.
↪format(bucket_name, prefix), content_type='csv')
```

```
's3_input' class will be renamed to 'TrainingInput' in SageMaker Python SDK v2.
```

[75]:
```
from sagemaker.amazon.amazon_estimator import get_image_uri
xgboost_container = get_image_uri(my_region,
                          'xgboost',
                          repo_version='1.2-1')

# Initialize hyperparameters
hyperparameters = {
        "max_depth":"5",
        "eta":"0.2",
        "alpha":"0.8",
        "min_child_weight":"6",
        "subsample":"0.7",
        "verbose":"1",
        "objective":"binary:logistic",
        "num-round":"100"}

xgb = sagemaker.estimator.Estimator(image_name = xgboost_container,
                hyperparameters=hyperparameters,
                role=role,
                train_instance_count=1,
                train_instance_type='ml.m4.xlarge',
                output_path='s3://{}/{}/output'.format(bucket_name, prefix))
```

```
'get_image_uri' method will be deprecated in favor of 'ImageURIProvider' class
in SageMaker Python SDK v2.
```

```
␣
↪---------------------------------------------------------------------------

ValueError                                Traceback (most recent call␣
↪last)

<ipython-input-75-855bcecd47a1> in <module>
  2 xgboost_container = get_image_uri(my_region,
  3                            'xgboost',
----> 4                            repo_version='1.2-1')
  5
  6 # Initialize hyperparameters


~/anaconda3/envs/python3/lib/python3.6/site-packages/sagemaker/amazon/
↪amazon_estimator.py in get_image_uri(region_name, repo_name, repo_version)
657             raise ValueError(
658                 "SageMaker XGBoost version {} is not supported.␣
↪Supported versions: {}".format(
--> 659                     repo_version, ", ".
↪join(XGBOOST_SUPPORTED_VERSIONS)
660                 )
661             )


ValueError: SageMaker XGBoost version 1.2-1 is not supported. Supported␣
↪versions: 0.90-1, 0.90-2, 1.0-1
```

[ ]:
```
xgb.fit({'train': s3_input_train})
```

[66]:
```
# Random Forest
from sagemaker import RandomCutForest
sess = sagemaker.Session()
# Setting hyper-parameters for the model
rcf = RandomCutForest(role = role,
                      train_instance_count=1,
                      train_instance_type='ml.m4.xlarge',
                      data_location='s3://{}/{}/train'.format(bucket_name,␣
 ↪prefix),
                      output_path='s3://{}/{}/output'.format(bucket_name,␣
 ↪prefix),
                      num_samples_per_tree=500,
                      num_trees=50)
```

[70]:
```
rcf.fit(rcf.record_set(train_data.value.to_numpy().reshape(-1,1)))

#rcf_predictor = rcf.deploy(initial_instance_count=1,instance_type='ml.m4.
 ↪xlarge')
```

```
      ␣
↪---------------------------------------------------------------------------

       AttributeError                            Traceback (most recent call␣
↪last)

       <ipython-input-70-dd5988cd6c99> in <module>
   ----> 1 rcf.fit(rcf.record_set(train_data.value.to_numpy().reshape(-1,1)))
         2
         3 #rcf_predictor = rcf.
↪deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')


       ~/anaconda3/envs/python3/lib/python3.6/site-packages/pandas/core/generic.
↪py in __getattr__(self, name)
      5272             if self._info_axis.
↪_can_hold_identifiers_and_holds_name(name):
      5273                 return self[name]
   -> 5274             return object.__getattribute__(self, name)
      5275
      5276     def __setattr__(self, name: str, value) -> None:


       AttributeError: 'DataFrame' object has no attribute 'value'
```