Approach for HCP Prediction (Identifying Healthcare Professionals and Taxonomy)

Problem Statement:

- We are provided with ad server logs of users that contain information like IP addresses, geographic locations, site URLs and related data.

- The task is to build a machine learning model using the ad server logs data and predict whether a user belongs to HCP (Healthcare Professionals) category or not.

- And if the user belongs to HCP category, we must predict the taxonomy of the HCP as well.

Approach for finding whether a user is Healthcare Professional (Predicting IS_HCP):

Step 1: Importing Data

Firstly, we import the train data and perform data analysis and pre-processing steps.

Step 2: Data Analysis and Pre-processing

We remove the data where IS_HCP value is null, Target column being null will not be useful for prediction.

One such row having ID=74791 is removed in this step.

Later, we study about each column and steps namely, null value identification and replacement, importance of the column, removing redundant columns.

Step 3: Feature Engineering

Depending on our study, we perform *Feature Engineering* such as, we create CIDRBLOCK and URLDOMAIN columns by leveraging data from BIDREQUESTIP and URL columns respectively.

For CIDRBLOCK, we find out the CIDR Notation Block depending on IP address from column BIDREQUESTIP. We categorize these CIDR Blocks into 5 classes. The method is as follows.

```python
# Method to calculate CIDR Block based on the ip block range as input

def Find_CIDR(ip_block):
    if ip_block >= 0 and ip_block <= 127:
        cidr_block = 0  # 'A' Block
    elif ip_block >= 128 and ip_block <= 191:
        cidr_block = 1  # 'B' Block
    elif ip_block >= 192 and ip_block <= 223:
        cidr_block = 2  # 'C' Block
    elif ip_block >=224 and ip_block <= 239:
        cidr_block = 3  # 'D' Block
    elif ip_block >= 240 and ip_block <= 255:
        cidr_block = 4  # 'E' Block
    else:
        cidr_block = np.nan

    return cidr_block
```

From URL column, we extract domain of the URL and categorize the values based on these domains.

We store these values in new column URLDOMAIN.

```python
# Method to calculate url_domain given url as input

def FindDomain(hcp_data):
    domains=[]
    for url in hcp_data:
        url_domain = url
        if ".org" in url:
            url_domain = url.split('.org')[0]
            url_domain += '.org/'
        else:
            url_domain = url.split('.com')[0]
            url_domain += '.com/'
        domains.append(url_domain)
    return domains
```

We create DEVICE column and try to find device type using USERAGENT column. We parse the User Agent String and extract few keywords, depending on that we decide the Device type.

```python
# Method to find Device Type through the User Agent String

def findDevice(ua_string):
    if 'smarttv' in ua_string:
        device = 'Smart TV'
    elif 'mobile' in ua_string or 'android' in ua_string or 'iphone' in ua_string:
        device = 'Mobile'
    elif 'ubuntu' in ua_string or 'windows' in ua_string or 'linux' in ua_string \
    or 'macintosh' in ua_string or 'cros'in ua_string:
        device = 'Desktop'
    elif 'tablet' in ua_string or 'ipad' in ua_string:
        device = 'Tablet'
    else:
        device = 'Unknown'
    return device
```

Step 4: Removing Redundant Columns / Feature Selection

We remove redundant columns such as 'ID', 'DEVICETYPE', 'BIDREQUESTIP', 'USERPLATFORMUID', 'USERCITY', 'USERZIPCODE', 'USERAGENT', 'CHANNELTYPE', 'TAXONOMY', 'PLATFORM_ID','URL','KEYWORDS'.

'CHANNELTYPE' contains only one value ‘Website’ so we remove this column. We don’t need ‘TAXONOMY’ for now, so removing that as well.

ID, USERPLATFORMUID, USERCITY, USERZIPCODE, KEYWORDS do not have any salient data patterns.

Information from columns DEVICETYPE, BIDREQUESTIP, PLATFORM_ID, URL has been leveraged.

So, we are left with these Final Feature columns: DEVICE, CIDRBLOCK, PLATFORMTYPE, URLDOMAIN.

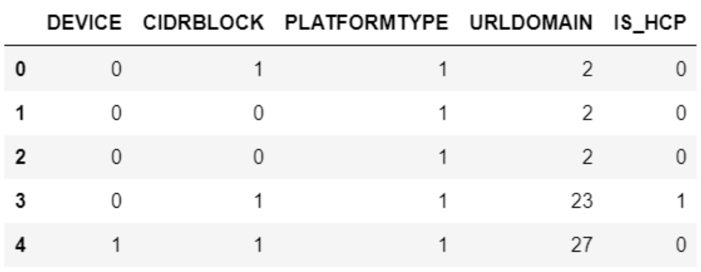
And our Target variable is: IS_HCP.

Step 5: Label Encoding Data

Starting with PLATFORMTYPE and URLDOMAIN, which have a certain relationship, we manually encode these columns, by creating unique values dictionary and assigning indexes.

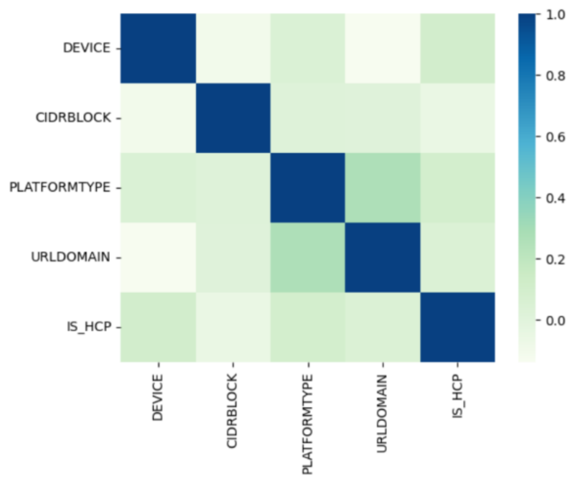
CIDRBLOCK is already in an encoded format. For DEVICE we use Scikit-Learn Label Encoder.

So, our Encoded Data is as follows:



	DEVICE	CIDRBLOCK	PLATFORMTYPE	URLDOMAIN	IS_HCP
0	0	1	1	2	0
1	0	0	1	2	0
2	0	0	1	2	0
3	0	1	1	23	1
4	1	1	1	27	0

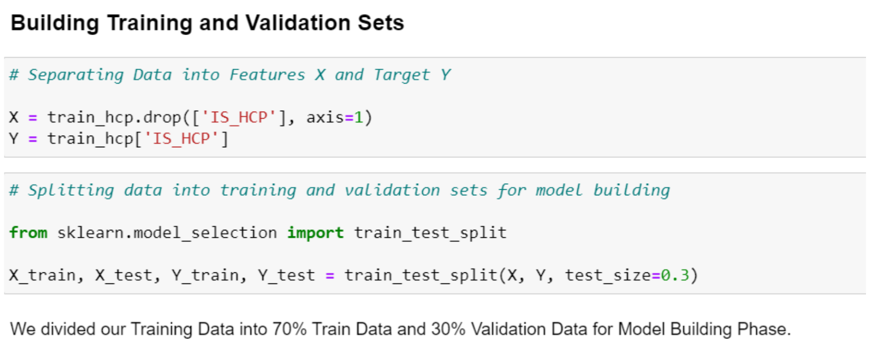
Following is the heatmap of correlations of features with target IS_HCP:



Step 6: Building Training and Validation Sets

Firstly, we separate our Features Data X and Target Y.

We use Train Test Split for creating Training and Validation Sets from X and Y datasets, using a 0.3 test size.



Building Training and Validation Sets

```
# Separating Data into Features X and Target Y

X = train_hcp.drop(['IS_HCP'], axis=1)
Y = train_hcp['IS_HCP']
```

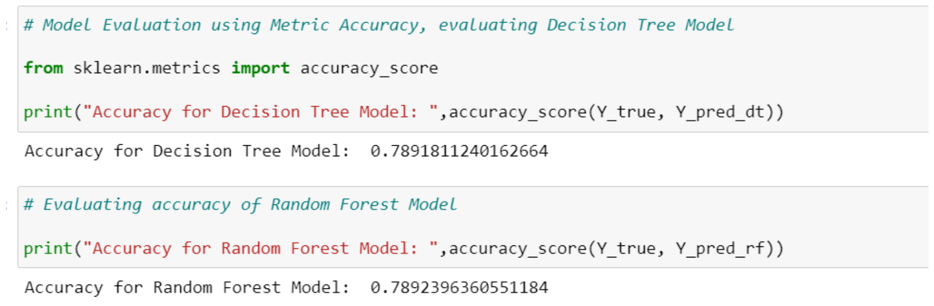
```
# Splitting data into training and validation sets for model building

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
```

We divided our Training Data into 70% Train Data and 30% Validation Data for Model Building Phase.

Step 7: Model Building (Model Name: HCP Model)

We choose 2 algorithms depending on data size, and speed: Decision Tree Algorithm and Random Forest Algorithm, and compare their accuracy performance.

Random Forest model outperforms, so we choose it as our Final Model.



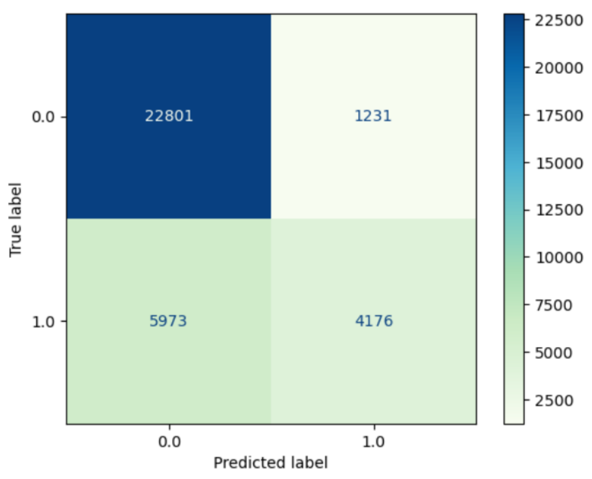
```
# Model Evaluation using Metric Accuracy, evaluating Decision Tree Model

from sklearn.metrics import accuracy_score

print("Accuracy for Decision Tree Model: ",accuracy_score(Y_true, Y_pred_dt))
```

```
Accuracy for Decision Tree Model:  0.7891811240162664
```

```
# Evaluating accuracy of Random Forest Model

print("Accuracy for Random Forest Model: ",accuracy_score(Y_true, Y_pred_rf))
```

```
Accuracy for Random Forest Model:  0.7892396360551184
```

Confusion Matrix of Final HCP Model:



Step 8: Prediction on Test Data

Finally, we import our test data, perform data transformation to make the data suitable for the model and get the predictions of whether the person is HCP or not.

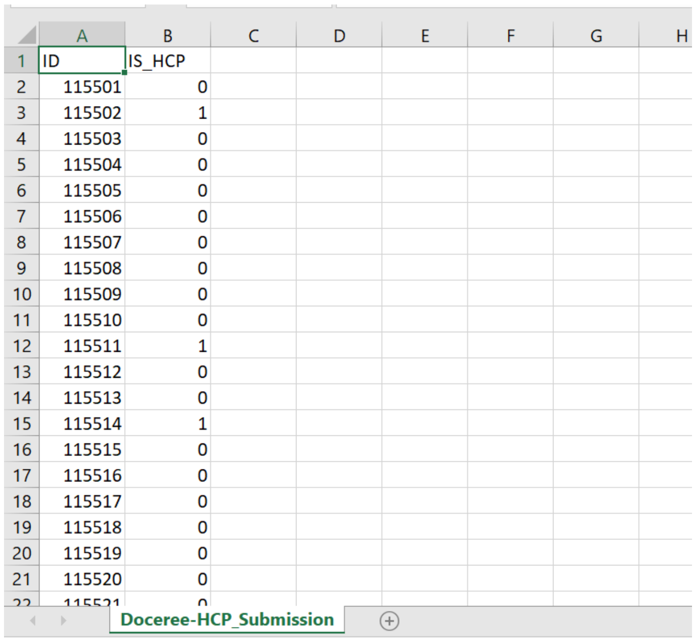
Data Transformation includes the following:

1) Creation of new columns CIDRBLOCK, URLDOMAIN, DEVICE
2) Removing Redundant Columns
3) Label Encoding Feature Data (Columns: DEVICE, PLATFORMTYPE, URLDOMAIN

We generate the predictions using our previously built HCP Model using Random Forest Algorithm.

Lastly, we store these predictions in a csv file named "Doceree-HCP_Submission.csv", following are the columns of the csv file:

1) ID: Key
2) IS_HCP: Indicates whether user is Healthcare Professional (HCP) or not



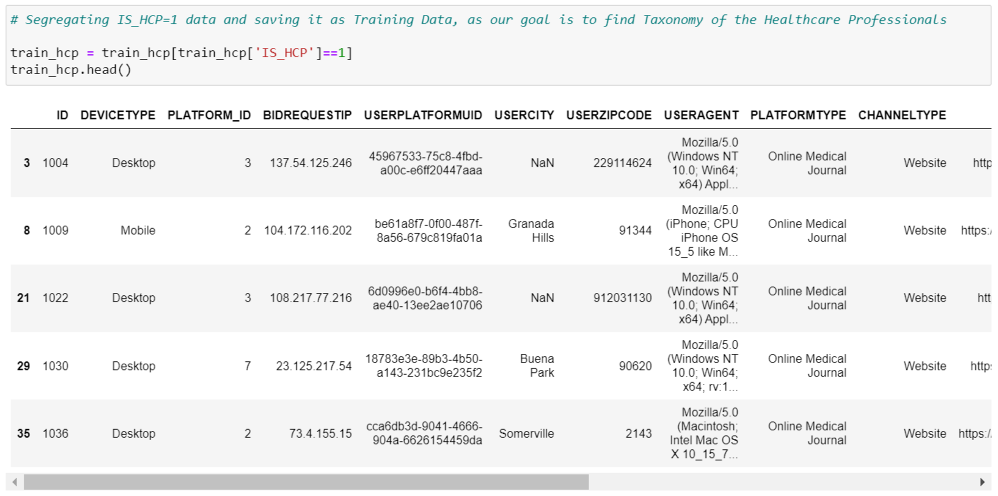
	A	B	C	D	E	F	G	H
1	ID	IS_HCP						
2	115501	0						
3	115502	1						
4	115503	0						
5	115504	0						
6	115505	0						
7	115506	0						
8	115507	0						
9	115508	0						
10	115509	0						
11	115510	0						
12	115511	1						
13	115512	0						
14	115513	0						
15	115514	1						
16	115515	0						
17	115516	0						
18	115517	0						
19	115518	0						
20	115519	0						
21	115520	0						
22	115521	0						

Approach for finding Taxonomy Healthcare Professional (Predicting Taxonomy):

Step 1: Importing Data

Firstly, we import the train data and perform data analysis and pre-processing steps.

We create a subset of training data where IS_HCP=1. As we must predict the Taxonomy, we will only require data of users who are HCP.



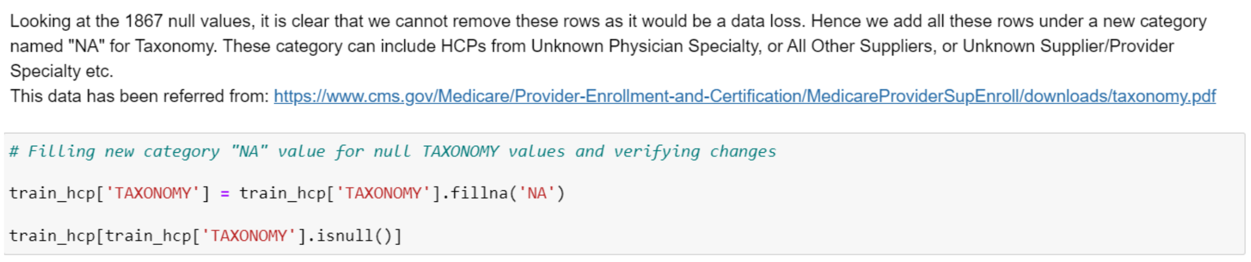
```
# Segregating IS_HCP=1 data and saving it as Training Data, as our goal is to find Taxonomy of the Healthcare Professionals

train_hcp = train_hcp[train_hcp['IS_HCP']==1]
train_hcp.head()
```

	ID	DEVICETYPE	PLATFORM_ID	BIDREQUESTIP	USERPLATFORMUID	USERCITY	USERZIPCODE	USERAGENT	PLATFORMTYPE	CHANNELTYPE	
3	1004	Desktop	3	137.54.125.246	45967533-75c8-4fbd-a00c-e6ff20447aaa	NaN	229114624	Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl...	Online Medical Journal	Website	http
8	1009	Mobile	2	104.172.116.202	be61a8f7-0f00-487f-8a56-679c819fa01a	Granada Hills	91344	Mozilla/5.0 (iPhone; CPU iPhone OS 15_5 like M...	Online Medical Journal	Website	https:/
21	1022	Desktop	3	108.217.77.216	6d0996e0-b6f4-4bb8-ae40-13ee2ae10706	NaN	912031130	Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl...	Online Medical Journal	Website	htt
29	1030	Desktop	7	23.125.217.54	18783e3e-89b3-4b50-a143-231bc9e235f2	Buena Park	90620	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:1...	Online Medical Journal	Website	http:
35	1036	Desktop	2	73.4.155.15	cca6db3d-9041-4666-904a-6626154459da	Somerville	2143	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7...	Online Medical Journal	Website	https:/

Step 2: Data Analysis and Pre-processing

We check all the null values from Target column TAXONOMY. In all we find 1867 null values.

These rows cannot be removed as it would lead to data loss. Hence, we introduce a new category NA for such entries, we fill up this category across all such entries.



Looking at the 1867 null values, it is clear that we cannot remove these rows as it would be a data loss. Hence we add all these rows under a new category named "NA" for Taxonomy. These category can include HCPs from Unknown Physician Specialty, or All Other Suppliers, or Unknown Supplier/Provider Specialty etc.
This data has been referred from: https://www.cms.gov/Medicare/Provider-Enrollment-and-Certification/MedicareProviderSupEnroll/downloads/taxonomy.pdf

```
# Filling new category "NA" value for null TAXONOMY values and verifying changes

train_hcp['TAXONOMY'] = train_hcp['TAXONOMY'].fillna('NA')

train_hcp[train_hcp['TAXONOMY'].isnull()]
```

Sources for Taxonomy Codes:

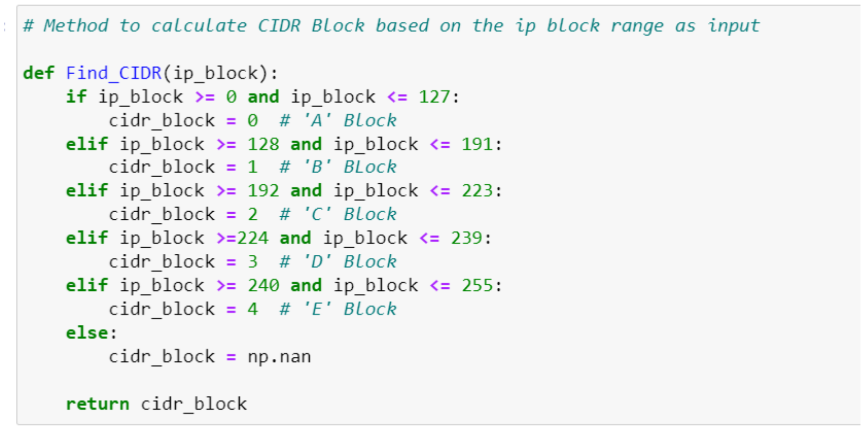
https://www.cms.gov/Medicare/Provider-Enrollment-and-Certification/MedicareProviderSupEnroll/downloads/taxonomy.pdf

https://nucc.org/images/stories/PDF/taxonomy_23_1.pdf

Step 3: Feature Engineering

Depending on our study, we perform *Feature Engineering* such as, we create CIDRBLOCK and URLDOMAIN columns by leveraging data from BIDREQUESTIP and URL columns respectively.

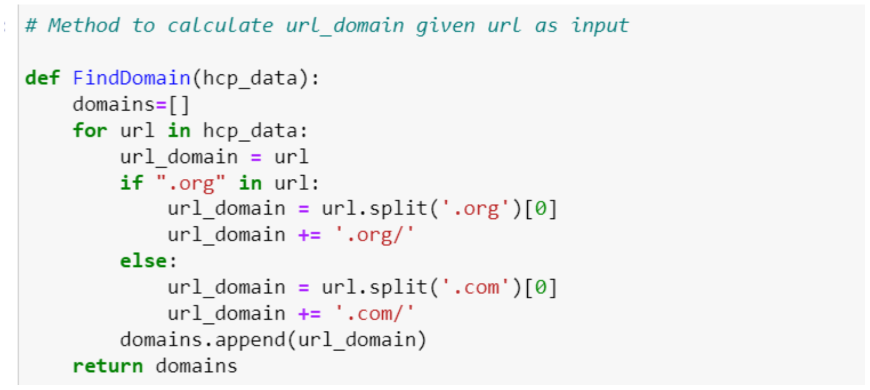
For CIDRBLOCK, we find out the CIDR Notation Block depending on IP address from column BIDREQUESTIP. We categorize these CIDR Blocks into 5 classes. The method is as follows.



```python
# Method to calculate CIDR Block based on the ip block range as input

def Find_CIDR(ip_block):
    if ip_block >= 0 and ip_block <= 127:
        cidr_block = 0  # 'A' Block
    elif ip_block >= 128 and ip_block <= 191:
        cidr_block = 1  # 'B' Block
    elif ip_block >= 192 and ip_block <= 223:
        cidr_block = 2  # 'C' Block
    elif ip_block >=224 and ip_block <= 239:
        cidr_block = 3  # 'D' Block
    elif ip_block >= 240 and ip_block <= 255:
        cidr_block = 4  # 'E' Block
    else:
        cidr_block = np.nan

    return cidr_block
```

From URL column, we extract domain of the URL and categorize the values based on these domains.

We store these values in new column URLDOMAIN.



```python
# Method to calculate url_domain given url as input

def FindDomain(hcp_data):
    domains=[]
    for url in hcp_data:
        url_domain = url
        if ".org" in url:
            url_domain = url.split('.org')[0]
            url_domain += '.org/'
        else:
            url_domain = url.split('.com')[0]
            url_domain += '.com/'
        domains.append(url_domain)
    return domains
```

Step 4: Removing Redundant Columns / Feature Selection

We remove redundant columns such as 'ID', 'BIDREQUESTIP', 'USERPLATFORMUID', 'USERCITY', 'USERZIPCODE', 'USERAGENT', 'CHANNELTYPE', IS_HCP, 'PLATFORM_ID','URL','KEYWORDS'.

'CHANNELTYPE' contains only one value ‘Website’ so we remove this column. We do not need ‘IS_HCP’ column, we have used it to get our data subset IS_HCP=1.

ID, USERPLATFORMUID, USERCITY, USERZIPCODE, KEYWORDS do not have any salient data patterns.

Information from columns BIDREQUESTIP, PLATFORM_ID, URL has been leveraged.

So, we are left with these Final Feature columns: DEVICETYPE, CIDRBLOCK, PLATFORMTYPE, URLDOMAIN.

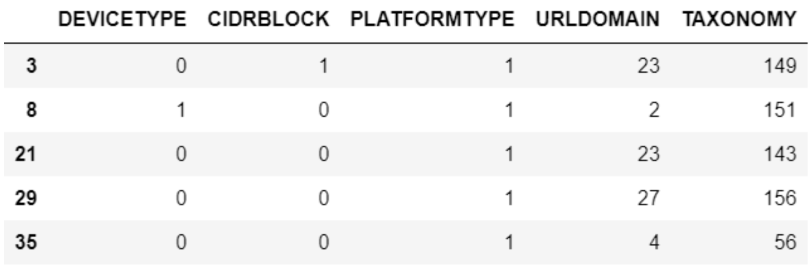
And our Target variable is: TAXONOMY.

Step 5: Label Encoding Data

Starting with PLATFORMTYPE and URLDOMAIN, which have a certain relationship, we manually encode these columns, by creating unique values dictionary and assigning indexes.

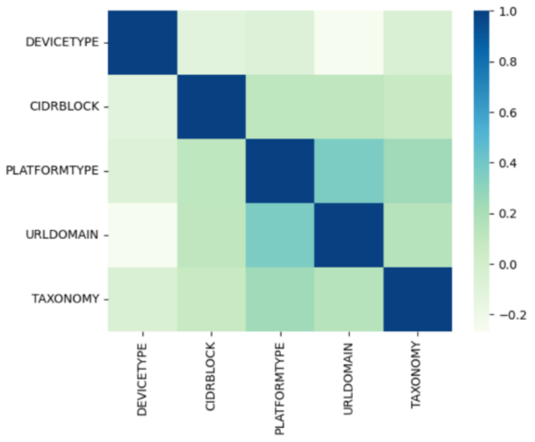
CIDRBLOCK is already in an encoded format. For DEVICETYPE and TAXONOMY, we use Scikit-Learn Label Encoder.

So, our Encoded Data is as follows:



	DEVICETYPE	CIDRBLOCK	PLATFORMTYPE	URLDOMAIN	TAXONOMY
3	0	1	1	23	149
8	1	0	1	2	151
21	0	0	1	23	143
29	0	0	1	27	156
35	0	0	1	4	56

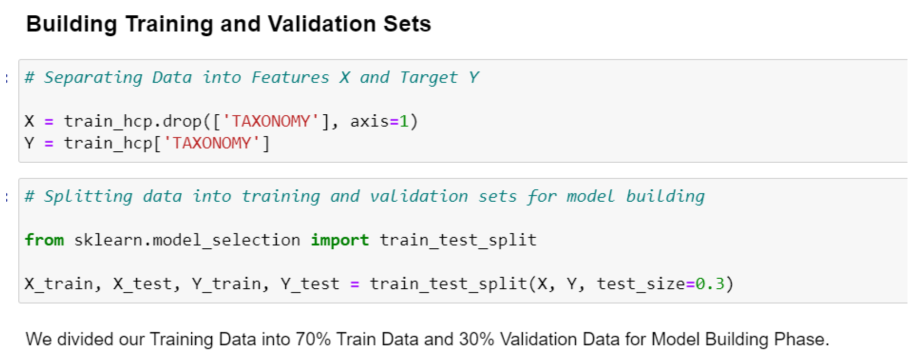
Following is the heatmap of correlations of features with target IS_HCP:



Step 6: Building Training and Validation Sets

Firstly, we separate our Features Data X and Target Y.

We use Train Test Split for creating Training and Validation Sets from X and Y datasets, using a 0.3 test size.



Building Training and Validation Sets

```
# Separating Data into Features X and Target Y

X = train_hcp.drop(['TAXONOMY'], axis=1)
Y = train_hcp['TAXONOMY']
```

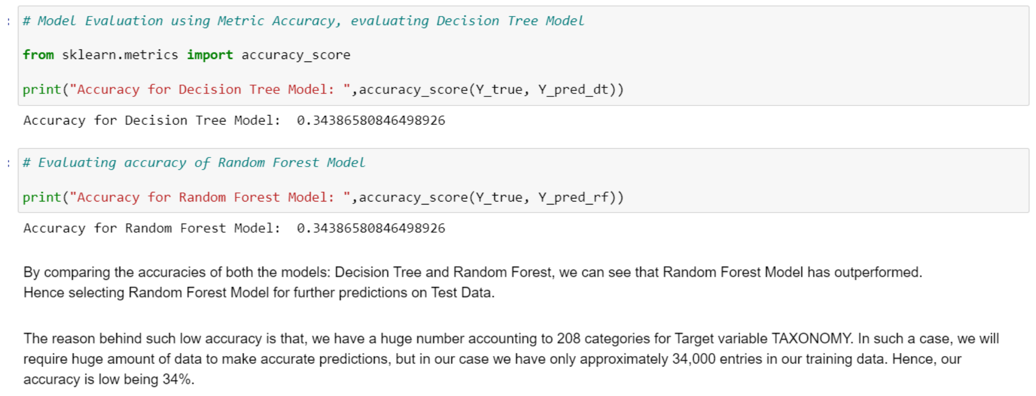
```
# Splitting data into training and validation sets for model building

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
```

We divided our Training Data into 70% Train Data and 30% Validation Data for Model Building Phase.

Step 7: Model Building (Model Name: Taxonomy Model)

We choose 2 algorithms depending on data size, and speed: Decision Tree Algorithm and Random Forest Algorithm, and compare their accuracy performance.

Random Forest model outperforms, so we choose it as our Final Model.



```
# Model Evaluation using Metric Accuracy, evaluating Decision Tree Model

from sklearn.metrics import accuracy_score

print("Accuracy for Decision Tree Model: ",accuracy_score(Y_true, Y_pred_dt))
```

```
Accuracy for Decision Tree Model:  0.34386580846498926
```

```
# Evaluating accuracy of Random Forest Model

print("Accuracy for Random Forest Model: ",accuracy_score(Y_true, Y_pred_rf))
```

```
Accuracy for Random Forest Model:  0.34386580846498926
```

By comparing the accuracies of both the models: Decision Tree and Random Forest, we can see that Random Forest Model has outperformed.
Hence selecting Random Forest Model for further predictions on Test Data.

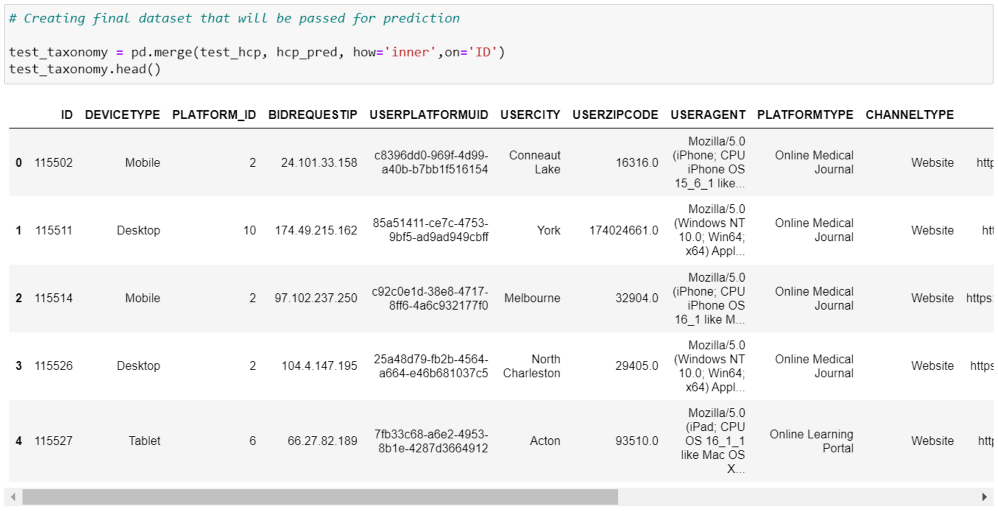
The reason behind such low accuracy is that, we have a huge number accounting to 208 categories for Target variable TAXONOMY. In such a case, we will require huge amount of data to make accurate predictions, but in our case we have only approximately 34,000 entries in our training data. Hence, our accuracy is low being 34%.

Step 8: Prediction on Test Data

Finally, we import our test data test_hcp and our HCP submission csv file hcp_submission. We create subset of hcp_submission data where IS_HCP=1. We call this subset as hcp_pred.

hcp_pred contains data that our HCP model has predicted. From that we segregate columns having IS_HCP as 1, i.e., our Model classifies these users as HCP, and hence we need to predict Taxonomy of such users.

We merge both these data frames, so that we get all the features data from test_hcp (Test Data) where our Model has predicted IS_HCP as 1.



```
# Creating final dataset that will be passed for prediction

test_taxonomy = pd.merge(test_hcp, hcp_pred, how='inner',on='ID')
test_taxonomy.head()
```

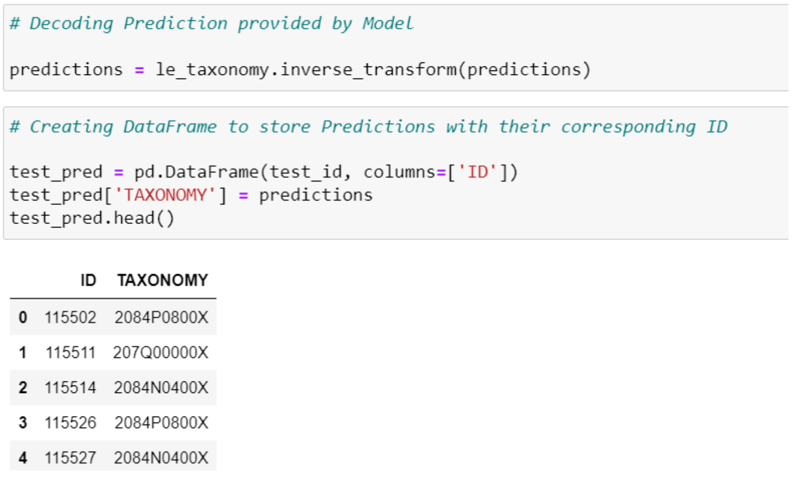
	ID	DEVICETYPE	PLATFORM_ID	BIDREQUESTIP	USERPLATFORMUID	USERCITY	USERZIPCODE	USERAGENT	PLATFORMTYPE	CHANNELTYPE	
0	115502	Mobile	2	24.101.33.158	c8396dd0-969f-4d99-a40b-b7bb1f516154	Conneaut Lake	16316.0	Mozilla/5.0 (iPhone; CPU iPhone OS 15_6_1 like...	Online Medical Journal	Website	http
1	115511	Desktop	10	174.49.215.162	85a51411-ce7c-4753-9bf5-ad9ad949cbff	York	174024661.0	Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl...	Online Medical Journal	Website	htt
2	115514	Mobile	2	97.102.237.250	c92c0e1d-38e8-4717-8ff6-4a6c932177f0	Melbourne	32904.0	Mozilla/5.0 (iPhone; CPU iPhone OS 16_1 like M...	Online Medical Journal	Website	https
3	115526	Desktop	2	104.4.147.195	25a48d79-fb2b-4564-a664-e46b681037c5	North Charleston	29405.0	Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl...	Online Medical Journal	Website	https
4	115527	Tablet	6	66.27.82.189	7fb33c68-a6e2-4953-8b1e-4287d3664912	Acton	93510.0	Mozilla/5.0 (iPad; CPU OS 16_1_1 like Mac OS X...	Online Learning Portal	Website	http

We perform Data Transformation on this test data as following:

4) Creation of new columns CIDRBLOCK, URLDOMAIN
5) Removing Redundant Columns
6) Label Encoding Feature Data (Columns: DEVICETYPE, PLATFORMTYPE, URLDOMAIN

We generate the predictions using our previously built Taxonomy Model using Random Forest Algorithm.

Next, we decode the Taxonomy predictions using Label Encoder Inverse Transform.



```
# Decoding Prediction provided by Model

predictions = le_taxonomy.inverse_transform(predictions)
```

```
# Creating DataFrame to store Predictions with their corresponding ID

test_pred = pd.DataFrame(test_id, columns=['ID'])
test_pred['TAXONOMY'] = predictions
test_pred.head()
```

	ID	TAXONOMY
0	115502	2084P0800X
1	115511	207Q00000X
2	115514	2084N0400X
3	115526	2084P0800X
4	115527	2084N0400X

Further, we merge our original HCP submission file contents with this predictions data frame, in order to obtain a combined result.

This combined result shows whether a user is HCP through IS_HCP column. If user is HCP, Taxonomy for the same is shown as well.

```python
# Merging Dataframes to show both IS_HCP and TAXONOMY Predictions

test_pred = pd.merge(test_pred, hcp_submission, how='outer', on='ID', sort=True)
test_pred.head()
```

	ID	TAXONOMY	IS_HCP
0	115501	NaN	0
1	115502	2084P0800X	1
2	115503	NaN	0
3	115504	NaN	0
4	115505	NaN	0

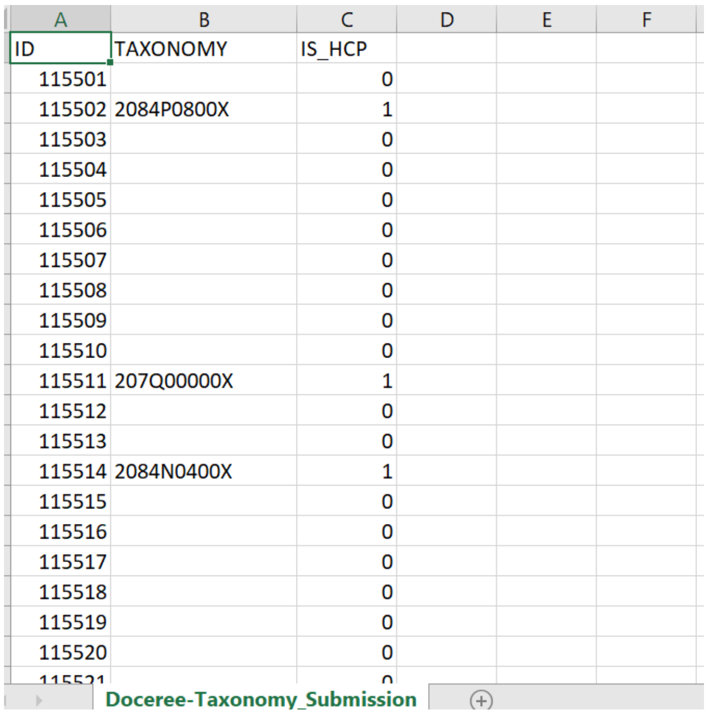
Note: Taxonomy will be NaN for IS_HCP=0. “NA”, our specially created category will be visible in this data frame for few instances IS_HCP=1.

Lastly, we store this combined result in a csv file named "Doceree-Taxonomy_Submission.csv", following are the columns of the csv file:

1) ID: Key
2) IS_HCP: Indicates whether user is Healthcare Professional (HCP) or not
3) Taxonomy: Specialization code for HCP

Taxonomy will be NaN for entries where user it not HCP i.e., IS_HCP=0.

Taxonomy “NA” specifies an Unspecified Taxonomy for entries where user is HCP but we do not know the Specialization.



A	B	C	D	E	F
ID	TAXONOMY	IS_HCP			
115501		0			
115502	2084P0800X	1			
115503		0			
115504		0			
115505		0			
115506		0			
115507		0			
115508		0			
115509		0			
115510		0			
115511	207Q00000X	1			
115512		0			
115513		0			
115514	2084N0400X	1			
115515		0			
115516		0			
115517		0			
115518		0			
115519		0			
115520		0			
115521		0			

Tools Used:

1) Jupyter Notebook
2) Microsoft Excel
3) Python Libraries such as Pandas, Numpy, Seaborn and Scikit-Learn

Source Files for Taxonomy Codes:

1) https://www.cms.gov/Medicare/Provider-Enrollment-and-Certification/MedicareProviderSupEnroll/downloads/taxonomy.pdf
2) https://nucc.org/images/stories/PDF/taxonomy_23_1.pdf

Rest all files have been provided during the hackathon.