Customer Segmentation Using IBM Watson

Machine Learning

1. INTRODUCTION:

1.1 Project Description:

In today's highly competitive world, the primal aim of any business is to grab potential customers who can generate profits for the organization. With increasing the number of organizations in the market, companies want to gain a competitive advantage over others.

The primal task of Management is to identify potential customers from the rest. This will be simplified with the help of Machine Learning models to classify the customers into segments based on various attributes.

The intervention of Data Science and AI helps the business to build such models to analyze the customers and their products in better decision making, to improvise the business process, to formulate better strategies, and to improve the revenue.

1.2 Purpose:

This project deals with understanding and segmenting the customers based on the data. The Model we built will be able to classify the customer's potentiality in purchasing power. We will be using classification algorithms such as H-clustering, k-means clustering Decision tree, Random Forest, KNN, and xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. Once the model is saved, we integrate it with the flask application and also deploy the model in IBM.

Customer Segmentation Using IBM Watson Machine Learning

2. Literature Survey:

a. Existing problem

Data quality issues also arise from a lack of maintenance and regular cleansing to ensure accuracy.

Another common problem with customer segmentation is that the business users do not understand the segmentation definitions and are using them incorrectly. There can be many customer segments set up to assist with specific business processes.
Obviously, we could create segments that differ demographically or in terms of behavior. But will these differences lead to a variety of marketing strategies that will be more effective than a single approach that treats all existing or potential customers the same way.

Intuitively, you know that your customer base, be it 50,000 or 5 million, is not homogeneous. In other words, at any point in time it will contain some good customers, some bad ones; some new customers, some old ones; some young, some old; some rich, some less rich, some poor; some price sensitive, some not; some extremely loyal, some not loyal at all.

b. Proposed Solution

Machine learning, a class of artificial intelligence, can investigate data sets of similar customers and interpret the most beneficial and most inadequate performing customer segments.

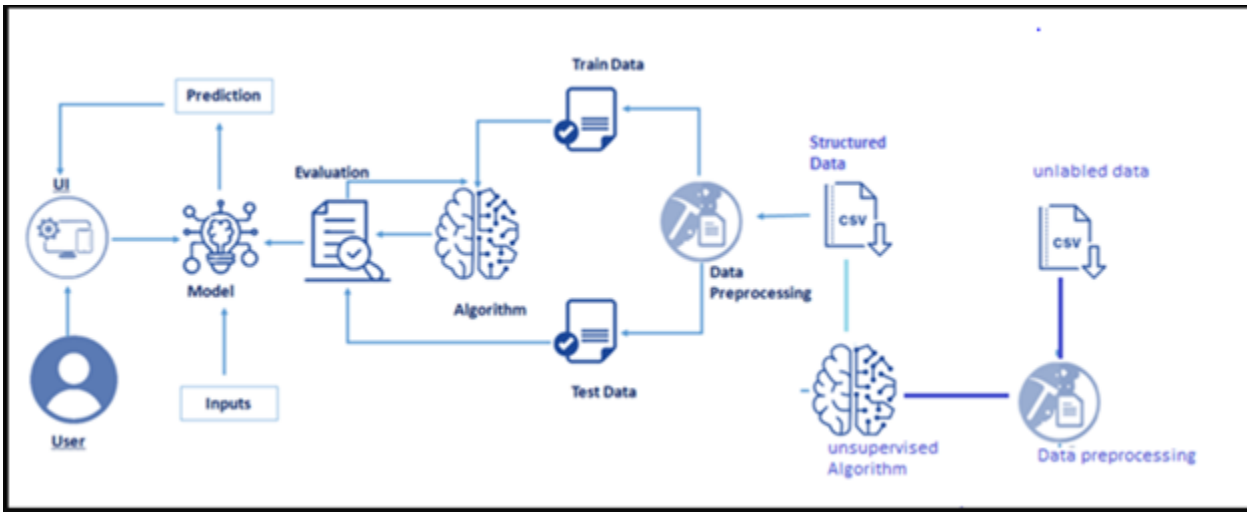
The subsequent actions are one of many strategies to tackle customer segmentation over machine learning. You can utilize your favorite tools, partners, and skills to handle these methods conveniently.
Customer segmentation is essential. Machine learning can get control over the complete process. Discovering all of the different groups that build up a more meaningful customer base permits you to get into customers' brains and give them precisely what they crave, enhancing their participation and expanding profits.

Customer Segmentation Using IBM Watson

Machine Learning

3. THEORITICAL ANALYSIS

a. Project Flow



b. Hardware / Software designing

Software Requirements:

To complete this project, you must require the following software's, concepts, and packages Anaconda navigator Python packages:

- numpy
- pandas.
- matplotlib.
- scikit-learn
- xgboost
- Flask

Hardware Requirements:

Customer Segmentation Using IBM Watson

Machine Learning

- Processor: Intel Core i3
- Hard Disk Space: Min 100 GB
- Ram: 4 GB
- Display: 14.1 “Color Monitor (LCD, CRT or LED)
- Clock Speed: 1.67 GHz

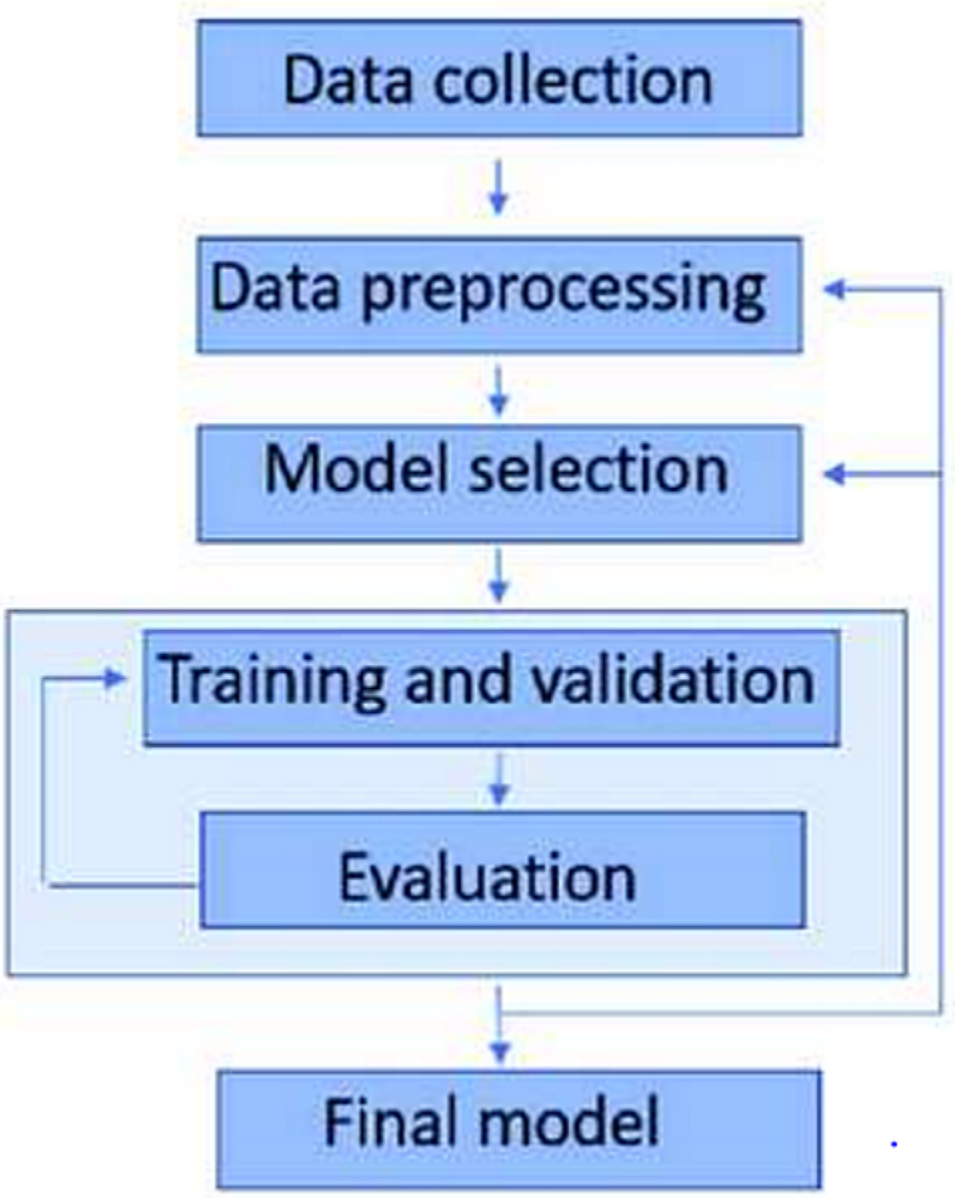
4. EXPERIMENTAL INVESTIGATIONS

ML depends heavily on data, without data, it is impossible for an “AI” model to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

Customer Segmentation Using IBM Watson

Machine Learning

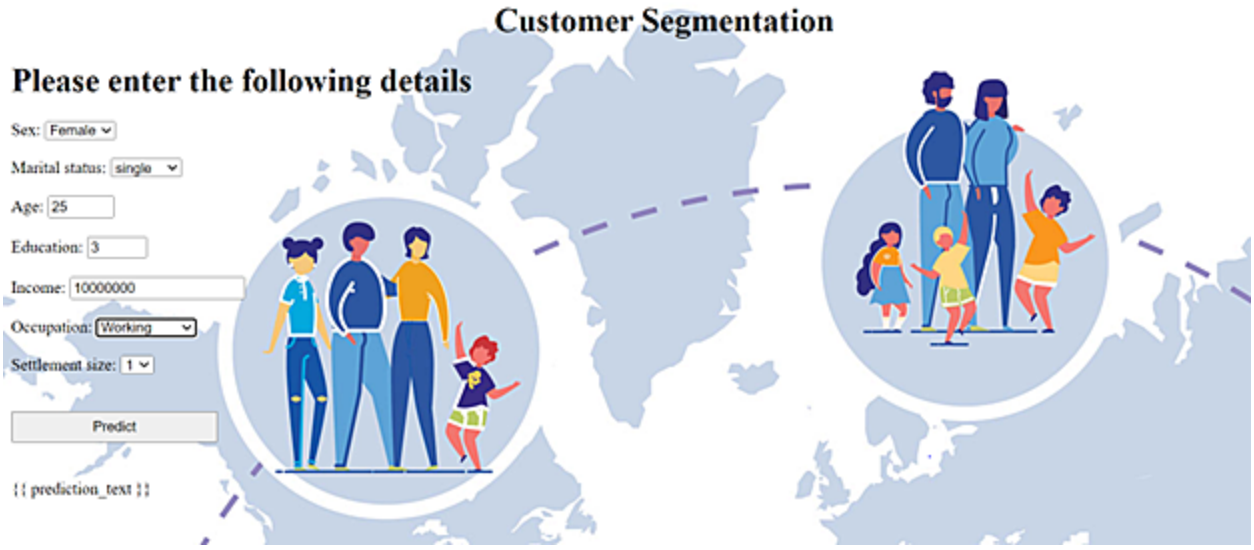
5. FLOWCHART



Customer Segmentation Using IBM Watson

Machine Learning

6. Result



Customer Segmentation Using IBM Watson

Machine Learning

7. Advantage & Disadvanatage:

- **Less time**

Manual customer segmentation is time-consuming. It takes months, even years to analyze piles of data and find patterns manually. Also if done heuristically, it may not have the accuracy to be useful as expected.

Customer segmentation used to be done manually and wasn't too precise. You'd manually create and populating different data tables, and analyze the data like a detective with a looking glass. Now, it's much better (and relatively easy thanks to rapid progress in ML) to just use machine learning, which can free up your time to focus on more demanding problems that require creativity to solve.

- **Ease of retraining**

Customer Segmentation is not a "develop once and use forever" type of project. Data is ever-changing, trends oscillate, everything keeps changing after your model is deployed. Usually, more labeled data becomes available after development, and it's a great resource for improving the overall performance of your model.

There are many ways to update customer segmentation models, but here are the two main approaches:

Use the old model as the starting point and retrain it.

Keep the existing model and combine its output with a new model.

- **Better scaling**

Machine learning models deployed in production support scalability, thanks to cloud infrastructure. These models are quite flexible for future changes and feedback. For example, consider a company that has 10000 customers today, and they've implemented a customer segmentation model. After a year, if the company has 1 million customers, then ideally we don't need to create a separate project to handle this increased data. Machine Learning models have the inherent capability to handle more data and scale in production.

- **Higher accuracy**

Customer Segmentation Using IBM Watson

Machine Learning

The value of an optimal number of clusters for given customer data is easy to find using machine learning methods like the elbow method. Not only the optimal number of clusters but also the performance of the model is far better when we use machine learning.

8. Applications

Implementing customer segmentation leads to plenty of new business opportunities. You can do a lot of optimizations in:

1. budgeting
2. product design
3. promotion
4. marketing
5. customer satisfaction.

9. Conclusion

It’s not wise to serve all customers with the same product model, email, text message campaign, or ad. Customers have different needs. A one-size-for-all approach to business will generally result in less engagement, lower-click through rates, and ultimately fewer sales. Customer segmentation is the cure for this problem.

Finding an optimal number of unique customer groups will help you understand how your customers differ, and help you give them exactly what they want. Customer segmentation improves customer experience and boosts company revenue. That’s why segmentation is a must if you want to surpass your competitors and get more customers. Doing it with machine learning is the right way to go.

Customer Segmentation Using IBM Watson Machine Learning

10. Bibliography

1. https://www.analyticsvidhya.com/blog/2021/06/how-to-solve-customer-segmentation-problem-with-machine-learning/
2. https://towardsdatascience.com/customer-segmentation-with-machine-learning-a0ac8c3d4d84

11. Appendix

a. Source Code

Customer Segmentation Using IBM Watson

Machine Learning

CLustering is the ML unsupervised methods,used to group the data based on similaries in the data - Hierarchial clustering

Hierarchial clustering

In [1]:
```
# similar records will be clubbed together
# DO EDA process - not mandatory(few are mandatory)
# scale the data
# calculate the distance - Euclidean or manhatten
# cluster the records based on single/complete link ('least /farthest ' distance,
# divide in the clusters into 2 or 3 classes based on the requirement
# use dendogram to visualise the clustered data
#join the classes with main data
```

In [2]:
```
import os
```

In [3]:
```
os.chdir('G:\AI&ML\ML projects\cluster analysis')
```

In [3]:
```
import pandas as pd
```

In [4]:
```
# Reading the dataset
data = pd.read_csv(r'E:\SmartBridge\Sathyabama mentoring\Customer segmentation m
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               2000 non-null   int64
 1   Sex              2000 non-null   int64
 2   Marital status   2000 non-null   int64
 3   Age              2000 non-null   int64
 4   Education        2000 non-null   int64
 5   Income           2000 non-null   int64
 6   Occupation       2000 non-null   int64
 7   Settlement size  2000 non-null   int64
dtypes: int64(8)
memory usage: 125.1 KB
```

Customer Segmentation Using IBM Watson

Machine Learning

In [6]:
```
data.shape
```

Out[6]: (2000, 8)

In [7]:
```
data.describe()
```

Out[7]:

	ID	Sex	Marital status	Age	Education	Income	Occupati
count	2.000000e+03	2000.000000	2000.000000	2000.000000	2000.00000	2000.000000	2000.0000
mean	1.000010e+08	0.457000	0.496500	35.909000	1.03800	120954.419000	0.8105
std	5.774946e+02	0.498272	0.500113	11.719402	0.59978	38108.824679	0.6385
min	1.000000e+08	0.000000	0.000000	18.000000	0.00000	35832.000000	0.0000
25%	1.000005e+08	0.000000	0.000000	27.000000	1.00000	97663.250000	0.0000
50%	1.000010e+08	0.000000	0.000000	33.000000	1.00000	115548.500000	1.0000
75%	1.000015e+08	1.000000	1.000000	42.000000	1.00000	138072.250000	1.0000
max	1.000020e+08	1.000000	1.000000	76.000000	3.00000	309364.000000	2.0000

In [8]:
```
cor = data.corr()
cor
```

Out[8]:

	ID	Sex	Marital status	Age	Education	Income	Occupation	Settleme si
ID	1.000000	0.328262	0.074403	-0.085246	0.012543	-0.303217	-0.291958	-0.3784
Sex	0.328262	1.000000	0.566511	-0.182885	0.244838	-0.195146	-0.202491	-0.3008
Marital status	0.074403	0.566511	1.000000	-0.213178	0.374017	-0.073528	-0.029490	-0.0970
Age	-0.085246	-0.182885	-0.213178	1.000000	0.654605	0.340610	0.108388	0.1197
Education	0.012543	0.244838	0.374017	0.654605	1.000000	0.233459	0.064524	0.0347
Income	-0.303217	-0.195146	-0.073528	0.340610	0.233459	1.000000	0.680357	0.4908
Occupation	-0.291958	-0.202491	-0.029490	0.108388	0.064524	0.680357	1.000000	0.5717
Settlement size	-0.378445	-0.300803	-0.097041	0.119751	0.034732	0.490881	0.571795	1.0000

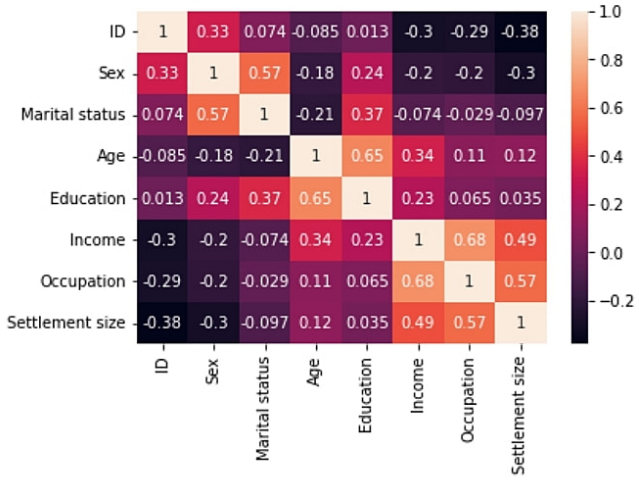
In [9]:
```
import seaborn as sns
```

Customer Segmentation Using IBM Watson

Machine Learning

In [10]:
```
sns.heatmap(cor,annot=True)
```

Out[10]: <AxesSubplot:>

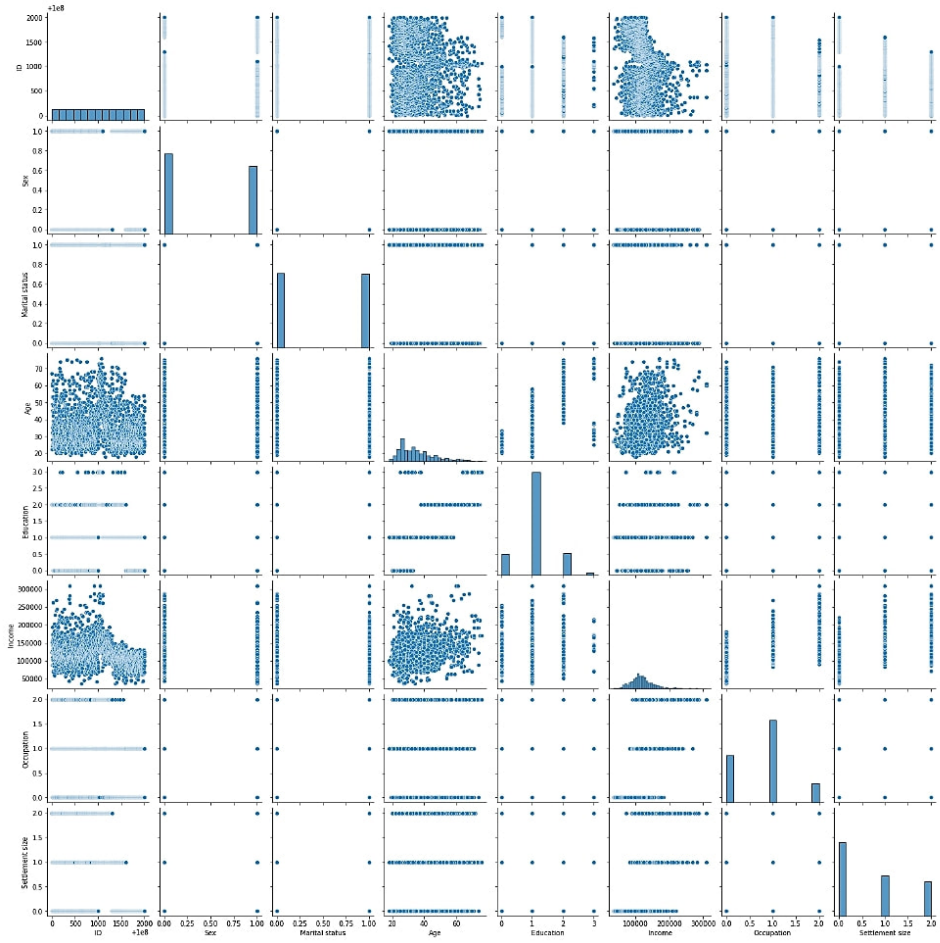


Customer Segmentation Using IBM Watson

Machine Learning

In [11]:
```
sns.pairplot(data)
```

Out[11]: <seaborn.axisgrid.PairGrid at 0x1d30ce4d7c0>



Customer Segmentation Using IBM Watson

Machine Learning

In [12]:
```
data.drop(columns=['ID'],axis=1,inplace=True)
```

In [13]:
```
data.head()
```

Out[13]:

	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
0	0	0	67	2	124670	1	2
1	1	1	22	1	150773	1	2
2	0	0	49	1	89210	0	0
3	0	0	45	1	171565	1	1
4	0	0	53	1	149031	1	1

In [14]:
```
names = data.columns
```

In [15]:
```
data.isna().sum()
```

Out[15]:
```
Sex                0
Marital status     0
Age                0
Education          0
Income             0
Occupation         0
Settlement size    0
dtype: int64
```

In [16]:
```
from sklearn import preprocessing
```

In [17]:
```
data = preprocessing.minmax_scale(data,feature_range=(0,1))  #scaling the data
```

Customer Segmentation Using IBM Watson

Machine Learning

In [18]:
```
data
```

Out[18]:
```
array([[0.        , 0.        , 0.84482759, ..., 0.32478101, 0.5       ,
        1.        ],
       [1.        , 1.        , 0.06896552, ..., 0.42021043, 0.5       ,
        1.        ],
       [0.        , 0.        , 0.53448276, ..., 0.19514353, 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.22413793, ..., 0.18487051, 0.        ,
        0.        ],
       [1.        , 1.        , 0.10344828, ..., 0.22716172, 0.        ,
        0.        ],
       [0.        , 0.        , 0.12068966, ..., 0.11912317, 0.        ,
        0.        ]])
```

In [19]:
```
data = pd.DataFrame(data,columns=names) #scaled data will convert to array,so cor
```

In [20]:
```
data.head()
```

Out[20]:

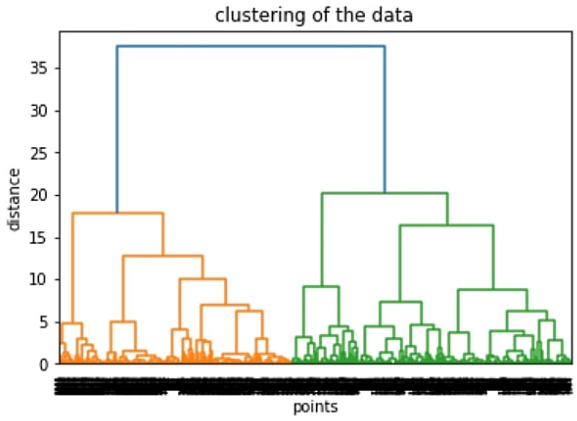
	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
0	0.0	0.0	0.844828	0.666667	0.324781	0.5	1.0
1	1.0	1.0	0.068966	0.333333	0.420210	0.5	1.0
2	0.0	0.0	0.534483	0.333333	0.195144	0.0	0.0
3	0.0	0.0	0.465517	0.333333	0.496223	0.5	0.5
4	0.0	0.0	0.603448	0.333333	0.413842	0.5	0.5

In [21]:
```
# using dendogram to find optimal no of clusters
import scipy.cluster.hierarchy as sch
import matplotlib as plt
```

Customer Segmentation Using IBM Watson

Machine Learning

In [22]:
```
dendogram = sch.dendrogram(sch.linkage(data,method="ward")) # calculates euclided
plt.pyplot.title('clustering of the data')
plt.pyplot.ylabel('distance')
plt.pyplot.xlabel('points')
plt.pyplot.show()
```



In [23]:
```
from sklearn import cluster
import sklearn as sk
```

In [24]:
```
clus = cluster.AgglomerativeClustering(n_clusters=3,affinity="euclidean",linkage=
clus
```

Out[24]:
```
AgglomerativeClustering(linkage='complete', n_clusters=3)
```

In [25]:
```
clus.fit(data)
```

Out[25]:
```
AgglomerativeClustering(linkage='complete', n_clusters=3)
```

In [26]:
```
abc = clus.fit_predict(data)
```

In [27]:
```
hclusdata = pd.DataFrame(data,pd.Series(abc))
```

Creating a labled data with the help of clustering model

In [28]:
```
hclusdata['clus']= pd.Series(abc)
```

Customer Segmentation Using IBM Watson

Machine Learning

In [29]:
```
hclusdata.head()
```

Out[29]:

	Sex	Marital status	Age	Education	Income	Occupation	Settlement size	clus
1	1.0	1.0	0.068966	0.333333	0.420210	0.5	1.0	0
0	0.0	0.0	0.844828	0.666667	0.324781	0.5	1.0	1
1	1.0	1.0	0.068966	0.333333	0.420210	0.5	1.0	0
1	1.0	1.0	0.068966	0.333333	0.420210	0.5	1.0	0
1	1.0	1.0	0.068966	0.333333	0.420210	0.5	1.0	0

In [30]:
```
y = hclusdata['clus']
x = hclusdata.drop(columns=['clus'],axis=1)
```

splitting the test and train data

In [31]:
```
from sklearn.model_selection import train_test_split
```

In [32]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=0
```

In [33]:
```
print(x_train.shape)
print(x_test.shape)
```

```
(1400, 7)
(600, 7)
```

Applying supervised learning on the data

In [34]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
import xgboost
```

In [35]:
```
rand_model = RandomForestClassifier()
tree_model =  tree.DecisionTreeClassifier()
xgb_model = xgboost.XGBClassifier()
```

Customer Segmentation Using IBM Watson

Machine Learning

In [36]:
```
rand_model.fit(x_train,y_train)
tree_model.fit(x_train,y_train)
xgb_model.fit(x_train,y_train)
```

Out[36]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

In [37]:
```
pred = rand_model.predict(x_test)
pred1 = tree_model.predict(x_test)
pred2 = xgb_model.predict(x_test)
```

In [38]:
```
from sklearn import metrics
```

In [39]:
```
print(metrics.accuracy_score(pred,y_test))
print(metrics.accuracy_score(pred1,y_test))
print(metrics.accuracy_score(pred2,y_test))
```

```
1.0
1.0
1.0
```

Customer Segmentation Using IBM Watson

Machine Learning

In [40]:
```
metrics.confusion_matrix(pred,y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_9404/3159782076.py in <module>
----> 1 metrics.confusion_matrix(pred,y_train)

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **k
wargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py in confusion_m
atrix(y_true, y_pred, labels, sample_weight, normalize)
    297
    298     """
--> 299     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    300     if y_type not in ("binary", "multiclass"):
    301         raise ValueError("%s is not supported" % y_type)

~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py in _check_targ
ets(y_true, y_pred)
     81     y_pred : array or indicator matrix
     82     """
---> 83     check_consistent_length(y_true, y_pred)
     84     type_true = type_of_target(y_true)
     85     type_pred = type_of_target(y_pred)

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_consistent_l
ength(*arrays)
    317     uniques = np.unique(lengths)
    318     if len(uniques) > 1:
--> 319         raise ValueError("Found input variables with inconsistent numbe
rs of"
    320                          " samples: %r" % [int(l) for l in lengths])
    321

ValueError: Found input variables with inconsistent numbers of samples: [600, 1
400]
```

K-means clustering

In [41]:
```
from scipy import spatial
```

Customer Segmentation Using IBM Watson

Machine Learning

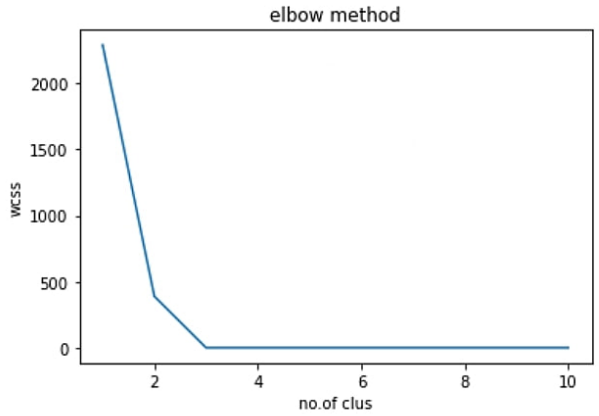
In [42]:
```
wcss = []
for i in range(1,11):
    kmeans = cluster.KMeans(n_clusters=i,init='k-means++',random_state=0)
    kmeans.fit(hclusdata)
    wcss.append(kmeans.inertia_)
```

```
C:\Users\lenovo\AppData\Local\Temp/ipykernel_9404/1710745755.py:4: ConvergenceW
arning: Number of distinct clusters (3) found smaller than n_clusters (4). Poss
ibly due to duplicate points in X.
  kmeans.fit(hclusdata)
C:\Users\lenovo\AppData\Local\Temp/ipykernel_9404/1710745755.py:4: ConvergenceW
arning: Number of distinct clusters (3) found smaller than n_clusters (5). Poss
ibly due to duplicate points in X.
  kmeans.fit(hclusdata)
C:\Users\lenovo\AppData\Local\Temp/ipykernel_9404/1710745755.py:4: ConvergenceW
arning: Number of distinct clusters (3) found smaller than n_clusters (6). Poss
ibly due to duplicate points in X.
  kmeans.fit(hclusdata)
C:\Users\lenovo\AppData\Local\Temp/ipykernel_9404/1710745755.py:4: ConvergenceW
arning: Number of distinct clusters (3) found smaller than n_clusters (7). Poss
ibly due to duplicate points in X.
  kmeans.fit(hclusdata)
C:\Users\lenovo\AppData\Local\Temp/ipykernel_9404/1710745755.py:4: ConvergenceW
arning: Number of distinct clusters (3) found smaller than n_clusters (8). Poss
ibly due to duplicate points in X.
  kmeans.fit(hclusdata)
C:\Users\lenovo\AppData\Local\Temp/ipykernel_9404/1710745755.py:4: ConvergenceW
arning: Number of distinct clusters (3) found smaller than n_clusters (9). Poss
ibly due to duplicate points in X.
  kmeans.fit(hclusdata)
C:\Users\lenovo\AppData\Local\Temp/ipykernel_9404/1710745755.py:4: ConvergenceW
arning: Number of distinct clusters (3) found smaller than n_clusters (10). Pos
sibly due to duplicate points in X.
  kmeans.fit(hclusdata)
```

Customer Segmentation Using IBM Watson

Machine Learning

In [43]:
```
plt.pyplot.plot(range(1,11),wcss)
plt.pyplot.title('elbow method')
plt.pyplot.xlabel('no.of clus')
plt.pyplot.ylabel('wcss')
plt.pyplot.show()
```



In [44]:
```
km_model = cluster.KMeans(n_clusters=3,init='k-means++',random_state=0)
```

In [45]:
```
ykmeans = km_model.fit_predict(data)
```

In [46]:
```
data.head()
```

Out[46]:

	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
0	0.0	0.0	0.844828	0.666667	0.324781	0.5	1.0
1	1.0	1.0	0.068966	0.333333	0.420210	0.5	1.0
2	0.0	0.0	0.534483	0.333333	0.195144	0.0	0.0
3	0.0	0.0	0.465517	0.333333	0.496223	0.5	0.5
4	0.0	0.0	0.603448	0.333333	0.413842	0.5	0.5

In [47]:
```
data['kclus'] = pd.Series(ykmeans)
```

Customer Segmentation Using IBM Watson

Machine Learning

In [48]:
```
data.head()
```

Out[48]:

	Sex	Marital status	Age	Education	Income	Occupation	Settlement size	kclus
0	0.0	0.0	0.844828	0.666667	0.324781	0.5	1.0	2
1	1.0	1.0	0.068966	0.333333	0.420210	0.5	1.0	1
2	0.0	0.0	0.534483	0.333333	0.195144	0.0	0.0	0
3	0.0	0.0	0.465517	0.333333	0.496223	0.5	0.5	2
4	0.0	0.0	0.603448	0.333333	0.413842	0.5	0.5	2

In [49]:
```
y = data['kclus']
x = data.drop(columns=['kclus'],axis=1)
```

In [50]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=
```

In [51]:
```
rand_model = RandomForestClassifier()
tree_model =  tree.DecisionTreeClassifier()
xgb_model = xgboost.XGBClassifier()
```

In [52]:
```
rand_model.fit(x_train,y_train)
tree_model.fit(x_train,y_train)
xgb_model.fit(x_train,y_train)
```

Out[52]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, objective='multi:softprob',
              predictor='auto', random_state=0, reg_alpha=0, ...)
```

In [53]:
```
pred = rand_model.predict(x_test)
pred1 = tree_model.predict(x_test)
pred2 = xgb_model.predict(x_test)
```

In [54]:
```
print(metrics.accuracy_score(pred,y_test))
print(metrics.accuracy_score(pred1,y_test))
print(metrics.accuracy_score(pred2,y_test))
```

```
0.9933333333333333
0.9983333333333333
0.9883333333333333
```

Saving the model

Customer Segmentation Using IBM Watson

Machine Learning

In [55]:
```
import pickle
```

In [56]:
```
pickle.dump(xgb_model,open("xgbmodel.pkl",'wb'))
```

In [57]:
```
pwd
```

Out[57]:
```
'E:\\SmartBridge\\Sathyabama mentoring\\Customer segmentation model\\Training'
```

In [ ]:
```

```

Customer Segmentation Using IBM Watson Machine Learning

App.py

```
import numpy as np

import pickle

import pandas

import os

from flask import Flask, request, jsonify, render_template

app = Flask(__name__)

model = pickle.load(open(r'xgbmodel.pkl', 'rb'))

#scale = pickle.load(open(r'C:/Users/SmartbridgePC/Desktop/AIML/Guided projects/rainfall_prediction/IBM flask push/Rainfall IBM deploy/scale.pkl','rb'))

@app.route('/')# route to display the home page

def home():

    return render_template('index.html') #rendering the home page

@app.route('/predict',methods=["POST","GET"])# route to show the predictions in a web UI

def predict():

    #  reading the inputs given by the user

    input_feature=[float(x) for x in request.form.values() ]

    features_values=[np.array(input_feature)]
```

Customer Segmentation Using IBM Watson Machine Learning

```
    names = [['Sex', 'Marital status', 'Age', 'Education', 'Income', 'Occupation',
       'Settlement size']]
    data = pandas.DataFrame(features_values,columns=names)
   # data = scale.fit_transform(features_values)
     # predictions using the loaded model file
    prediction=model.predict(data)
    print(prediction)
    if (prediction == 0):
       return render_template("notimp.html",prediction_text ="Not a potential customer")
    elif (prediction == 1):
       return render_template("imp.html",prediction_text = "Potential customer")
    else:
       return render_template("moreimp.html",prediction_text = "Highly potential customer")
     # showing the prediction results in a UI
if __name__=="__main__":
    # app.run(host='0.0.0.0', port=8000,debug=True)    # running the app
    port=int(os.environ.get('PORT',5000))
    app.run(port=port,debug=True,use_reloader=False)
```