ML FSN Programming

Linear Regression (CO2 Emissions)

```
import seaborn as sns
import pandas as pd
import numpy as np
```

Downloading Data

```
!wget -O FuelConsumption.csv https://cf-courses-data.s3.us.cloud-object-storage.appdoma
```

```
--2022-06-20 08:54:07--  https://cf-courses-data.s3.us.cloud-object-storage.appdomain
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-
HTTP request sent, awaiting response... 200 OK
Length: 72629 (71K) [text/csv]
Saving to: ‘FuelConsumption.csv’

FuelConsumption.csv 100%[===================>]  70.93K  --.-KB/s    in 0.04s

2022-06-20 08:54:07 (1.93 MB/s) - ‘FuelConsumption.csv’ saved [72629/72629]
```

Reading the data in

```
df = pd.read_csv("FuelConsumption.csv")

# take a look at the dataset
df.head()
```

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FU
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5	

```
df.corr()
```

	MODELYEAR	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CI
MODELYEAR	NaN	NaN	NaN	N
ENGINESIZE	NaN	1.000000	0.934011	0.8322
CYLINDERS	NaN	0.934011	1.000000	0.7964
FUELCONSUMPTION_CITY	NaN	0.832225	0.796473	1.0000
FUELCONSUMPTION_HWY	NaN	0.778746	0.724594	0.9657
FUELCONSUMPTION_COMB	NaN	0.819482	0.776788	0.9955
FUELCONSUMPTION_COMB_MPG	NaN	-0.808554	-0.770430	-0.9356
CO2EMISSIONS	NaN	0.874154	0.849685	0.8980

```
df.corr()['CO2EMISSIONS'].sort_values(ascending=False)
```

```
CO2EMISSIONS                1.000000
FUELCONSUMPTION_CITY        0.898039
FUELCONSUMPTION_COMB        0.892129
ENGINESIZE                  0.874154
FUELCONSUMPTION_HWY         0.861748
CYLINDERS                   0.849685
FUELCONSUMPTION_COMB_MPG   -0.906394
MODELYEAR                        NaN
Name: CO2EMISSIONS, dtype: float64
```

Let's select some features that we want to use for regression.

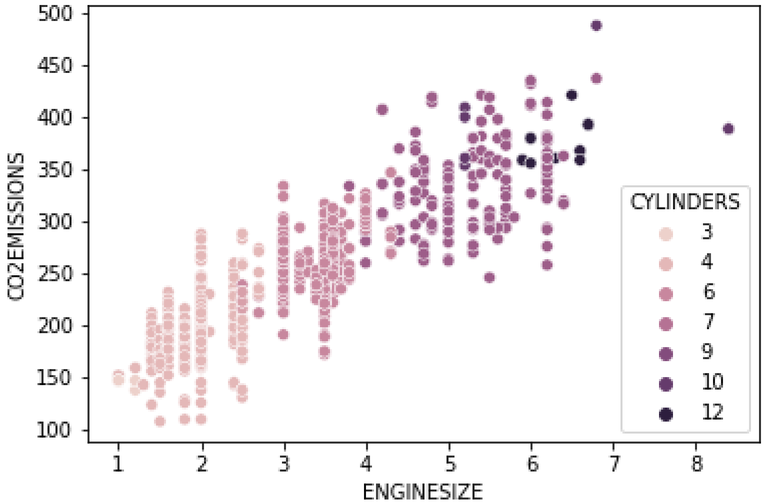
```
cdf = df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_CITY','FUELCONSUMPTION_HWY','FUELCO
cdf.head(9)
```

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY	FUELCONSUMPTION_
0	2.0	4	9.9	6.7	
1	2.4	4	11.2	7.7	
2	1.5	4	6.0	5.8	

Let's plot Emission values with respect to Engine size:

```
import seaborn as sns
sns.scatterplot(data=cdf,x=cdf.ENGINESIZE,y=cdf.CO2EMISSIONS,hue=cdf.CYLINDERS)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff42d6a6a90>
```




```
cdf.columns
```

```
Index(['ENGINESIZE', 'CYLINDERS', 'FUELCONSUMPTION_CITY',
       'FUELCONSUMPTION_HWY', 'FUELCONSUMPTION_COMB', 'CO2EMISSIONS'],
      dtype='object')
```

```
cdf_x=cdf[['ENGINESIZE', 'CYLINDERS','FUELCONSUMPTION_COMB']]
```

```
cdf_y=cdf[['CO2EMISSIONS']]
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(cdf_x,cdf_y,test_size=0.2,random_state=4
```

```
X_train.head()
```

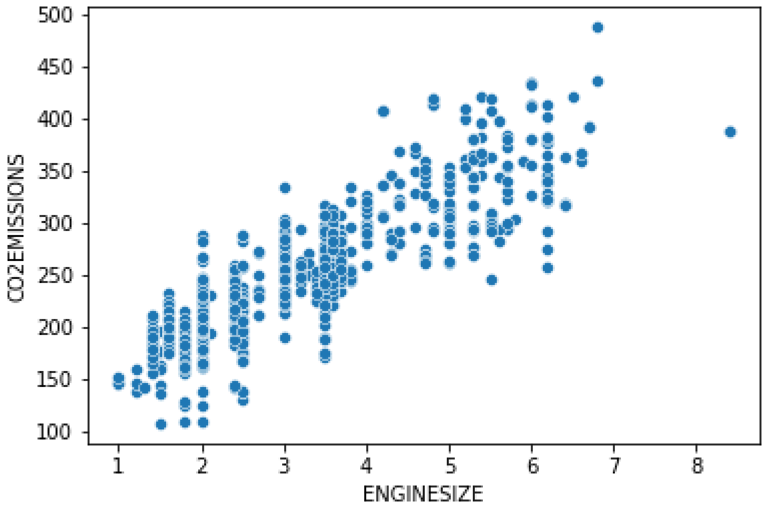
ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB

```
1 y_train.head()
```

	CO2EMISSIONS
409	196
773	205
146	264
776	246
381	336

```
1 sns.scatterplot(X_train.ENGINESIZE,y_train.CO2EMISSIONS)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7ff42b982d90>
```



Multiple Regression Model

```
1 from sklearn import linear_model
2 regr = linear_model.LinearRegression()
3 regr
```

```
LinearRegression()
```

```
1 regr.fit (X_train, y_train)
```

```
LinearRegression()
```

```
1 regr.intercept_
```

```
array([65.2578757])
```

```
regr.coef_
```

```
array([[10.24537129,  7.64355532,  9.68132732]])
```

```

```

Prediction

```
y_hat= regr.predict(X_test)
y_hat[0:5]
```

```
array([[259.39421287],
       [216.04051098],
       [255.40887315],
       [261.21766954],
       [294.43880893]])
```

```
x = X_test
y = y_test
```

```
np.mean(y_hat-y)
```

```
CO2EMISSIONS   -1.375796
dtype: float64
```

```
#Residual sum of squares:
np.mean((y_hat - y) ** 2)
```

```
CO2EMISSIONS    408.37553
dtype: float64
```

```
# Explained variance score: 1 is perfect prediction
#print('Variance score: %.2f' %
regr.score(x, y)
```

```
0.890023090970219
```

KNN (Customer Category)

```
!wget -O teleCust1000t.csv https://cf-courses-data.s3.us.cloud-object-storage.appdomain
```

```
--2022-06-20 09:03:33--  https://cf-courses-data.s3.us.cloud-object-storage.appdomain
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-
```

```
HTTP request sent, awaiting response... 200 OK
Length: 36047 (35K) [text/csv]
Saving to: ‘teleCust1000t.csv’

teleCust1000t.csv   100%[===================>]  35.20K  --.-KB/s    in 0.02s

2022-06-20 09:03:33 (1.39 MB/s) - ‘teleCust1000t.csv’ saved [36047/36047]
```

```
df = pd.read_csv('teleCust1000t.csv')
df.head()
```

	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside
0	2	13	44	1	9	64.0	4	5	0.0	0	2
1	3	11	33	1	7	136.0	5	5	0.0	0	6
2	3	68	52	1	24	116.0	1	29	0.0	1	2
3	2	33	33	0	12	33.0	2	0	0.0	1	1
4	2	23	30	1	9	30.0	1	2	0.0	0	4

```
df.custcat.value_counts()
```

```
3    281
1    266
4    236
2    217
Name: custcat, dtype: int64
```

```
df.corr()['custcat'].sort_values(ascending=False)
```

```
custcat    1.000000
ed         0.193864
tenure     0.166691
income     0.134525
employ     0.110011
marital    0.083836
reside     0.082022
address    0.067913
age        0.056909
retire     0.008908
gender    -0.004966
region    -0.023771
Name: custcat, dtype: float64
```

```
df.columns
```

```
Index(['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed',
       'employ', 'retire', 'gender', 'reside', 'custcat'],
      dtype='object')
```

Normalize Data Data Standardization gives the data zero mean and unit variance, it is good practice, especially for algorithms such as KNN which is based on the distance of data points:

TO use scikit learn convert pandas df to np array

```
X = df[['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed','employ', 'reti
X[0:5]
```

	region	tenure	age	marital	address	income	ed	employ	retire	gender	reside
0	2	13	44	1	9	64.0	4	5	0.0	0	2
1	3	11	33	1	7	136.0	5	5	0.0	0	6
2	3	68	52	1	24	116.0	1	29	0.0	1	2
3	2	33	33	0	12	33.0	2	0	0.0	1	1
4	2	23	30	1	9	30.0	1	2	0.0	0	4

its still df, so use .values

```
X = df[['region', 'tenure', 'age', 'marital', 'address', 'income', 'ed','employ', 'reti
X[0:5]
```

```
array([[  2.,  13.,  44.,   1.,   9.,  64.,   4.,   5.,   0.,   0.,   2.],
       [  3.,  11.,  33.,   1.,   7., 136.,   5.,   5.,   0.,   0.,   6.],
       [  3.,  68.,  52.,   1.,  24., 116.,   1.,  29.,   0.,   1.,   2.],
       [  2.,  33.,  33.,   0.,  12.,  33.,   2.,   0.,   0.,   1.,   1.],
       [  2.,  23.,  30.,   1.,   9.,  30.,   1.,   2.,   0.,   0.,   4.]])
```

```
Y = df[['custcat']].values
Y[0:5]
```

```
array([[1],
       [4],
       [3],
       [1],
       [3]])
```

```
X.shape
```

```
(1000, 11)
```

```
Y.shape
```

```
(1000, 1)
```

```
from sklearn import preprocessing
```

```
X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
X[0:5]
```

```
array([[-0.02696767, -1.055125  ,  0.18450456,  1.0100505 , -0.25303431,
        -0.12650641,  1.0877526 , -0.5941226 , -0.22207644, -1.03459817,
        -0.23065004],
       [ 1.19883553, -1.14880563, -0.69181243,  1.0100505 , -0.4514148 ,
         0.54644972,  1.9062271 , -0.5941226 , -0.22207644, -1.03459817,
         2.55666158],
       [ 1.19883553,  1.52109247,  0.82182601,  1.0100505 ,  1.23481934,
         0.35951747, -1.36767088,  1.78752803, -0.22207644,  0.96655883,
        -0.23065004],
       [-0.02696767, -0.11831864, -0.69181243, -0.9900495 ,  0.04453642,
        -0.41625141, -0.54919639, -1.09029981, -0.22207644,  0.96655883,
        -0.92747794],
       [-0.02696767, -0.58672182, -0.93080797,  1.0100505 , -0.25303431,
        -0.44429125, -1.36767088, -0.89182893, -0.22207644, -1.03459817,
         1.16300577]])
```

```
X.shape,Y.shape
```

```
((1000, 11), (1000, 1))
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, Y_train , X_test , Y_test = train_test_split(X,Y,test_size=0.2,random_state=4)
#order is wrong , X train then X_test
```

```
X_train.shape , Y_train.shape
```

```
((800, 11), (200, 11))
```

```
X_train, X_test, Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=4)
```

```
X_train.shape , Y_train.shape
```

```
((800, 11), (800, 1))
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
k=4
KNN = KNeighborsClassifier(n_neighbors = k).fit(X_train,Y_train)
KNN
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
KNeighborsClassifier(n_neighbors=4)
```

```
Y_hat = KNN.predict(X_test)
```

```
Y_hat[0:5]
```

```
array([1, 1, 3, 2, 4])
```

for multilabel classifier , we use jaccard score

```
from sklearn import metrics
```

```
metrics.accuracy_score(Y_test,Y_hat)
```

```
0.32
```

```
metrics.accuracy_score(Y_hat,Y_test)
```

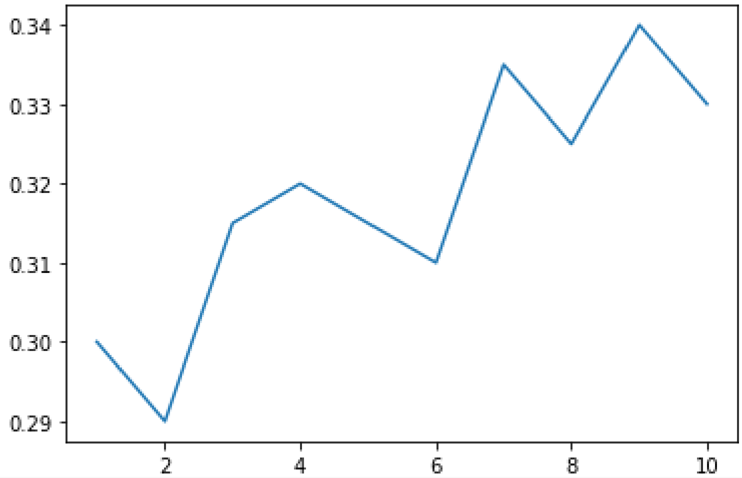
```
0.32
```

```
k_max = 11
k_score_l=[]
for i in range(1,k_max):
    KNN = KNeighborsClassifier(n_neighbors=i).fit(X_train,Y_train)
    Y_hat = KNN.predict(X_test)
    scr = metrics.accuracy_score(Y_test,Y_hat)
    k_score_l.append(scr)
k_score_l
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: Data
  return self._fit(X, y)
[0.3, 0.29, 0.315, 0.32, 0.315, 0.31, 0.335, 0.325, 0.34, 0.33]
```

```
sns.lineplot(y=np.array(k_score_l),x=np.arange(1,11))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff42178d090>
```




```
# k=9 is max score , do that score
```

Decision Tree (Drug A B C)

```
!wget -O drug200.csv https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud
```

```
--2022-06-20 11:26:40--  https://cf-courses-data.s3.us.cloud-object-storage.appdomain
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-
HTTP request sent, awaiting response... 200 OK
Length: 5827 (5.7K) [text/csv]
Saving to: ‘drug200.csv’

drug200.csv         100%[===================>]   5.69K  --.-KB/s    in 0s

2022-06-20 11:26:40 (1.16 GB/s) - ‘drug200.csv’ saved [5827/5827]
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
drug = pd.read_csv("drug200.csv")
drug
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY

```
drug.head()
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY

```
X = drug[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values
X[0:5]
```

```
array([[23, 'F', 'HIGH', 'HIGH', 25.355],
       [47, 'M', 'LOW', 'HIGH', 13.093],
       [47, 'M', 'LOW', 'HIGH', 10.114],
       [28, 'F', 'NORMAL', 'HIGH', 7.798],
       [61, 'F', 'LOW', 'HIGH', 18.043]], dtype=object)
```

```
Y = drug[['Drug']].values
Y[0:5]
```

```
array([['drugY'],
       ['drugC'],
       ['drugC'],
       ['drugX'],
       ['drugY']], dtype=object)
```

```
drug.columns
```

```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

```
from sklearn import preprocessing


LE_Sex = preprocessing.LabelEncoder()
LE_Sex.fit(['M','F'])
X[:,1]= LE_Sex.transform(X[:,1])
X[:5]
```

```
array([[23, 0, 'HIGH', 'HIGH', 25.355],
       [47, 1, 'LOW', 'HIGH', 13.093],
       [47, 1, 'LOW', 'HIGH', 10.114],
       [28, 0, 'NORMAL', 'HIGH', 7.798],
       [61, 0, 'LOW', 'HIGH', 18.043]], dtype=object)
```

```
drug.BP.unique()
```

```
array(['HIGH', 'LOW', 'NORMAL'], dtype=object)
```

```
LE_BP = preprocessing.LabelEncoder()
LE_BP.fit(['HIGH', 'LOW', 'NORMAL'])
X[:,2]=LE_BP.transform(X[:,2])
X[:5]
```

```
array([[23, 0, 0, 'HIGH', 25.355],
       [47, 1, 1, 'HIGH', 13.093],
       [47, 1, 1, 'HIGH', 10.114],
       [28, 0, 2, 'HIGH', 7.798],
       [61, 0, 1, 'HIGH', 18.043]], dtype=object)
```

```
drug.Cholesterol.unique()
```

```
array(['HIGH', 'NORMAL'], dtype=object)
```

```
LE_Chol = preprocessing.LabelEncoder()
LE_Chol.fit([ 'NORMAL', 'HIGH'])
X[:,3] = LE_Chol.transform(X[:,3])
```

```
X
```

```
array([[23, 0, 0, 0, 25.355],
       [47, 1, 1, 0, 13.093],
       [47, 1, 1, 0, 10.114],
       [28, 0, 2, 0, 7.798],
       [61, 0, 1, 0, 18.043],
       [22, 0, 2, 0, 8.607],
       [49, 0, 2, 0, 16.275],
       [41, 1, 1, 0, 11.037],
       [60, 1, 2, 0, 15.171],
       [43, 1, 1, 1, 19.368],
       [47, 0, 1, 0, 11.767],
       [34, 0, 0, 1, 19.199],
       [43, 1, 1, 0, 15.376],
       [74, 0, 1, 0, 20.942],
       [50, 0, 2, 0, 12.703],
       [16, 0, 0, 1, 15.516],
       [69, 1, 1, 1, 11.455],
       [43, 1, 0, 0, 13.972],
       [23, 1, 1, 0, 7.298],
       [32, 0, 0, 1, 25.974],
       [57, 1, 1, 1, 19.128],
       [63, 1, 2, 0, 25.917],
       [47, 1, 1, 1, 30.568],
```

```
       [48, 0, 1, 0, 15.036],
       [33, 0, 1, 0, 33.486],
       [28, 0, 0, 1, 18.809],
       [31, 1, 0, 0, 30.366],
       [49, 0, 2, 1, 9.381],
       [39, 0, 1, 1, 22.697],
       [45, 1, 1, 0, 17.951],
       [18, 0, 2, 1, 8.75],
       [74, 1, 0, 0, 9.567],
       [49, 1, 1, 1, 11.014],
       [65, 0, 0, 1, 31.876],
       [53, 1, 2, 0, 14.133],
       [46, 1, 2, 1, 7.285],
       [32, 1, 0, 1, 9.445],
       [39, 1, 1, 1, 13.938],
       [39, 0, 2, 1, 9.709],
       [15, 1, 2, 0, 9.084],
       [73, 0, 2, 0, 19.221],
       [58, 0, 0, 1, 14.239],
       [50, 1, 2, 1, 15.79],
       [23, 1, 2, 0, 12.26],
       [50, 0, 2, 1, 12.295],
       [66, 0, 2, 1, 8.107],
       [37, 0, 0, 0, 13.091],
       [68, 1, 1, 0, 10.291],
       [23, 1, 2, 0, 31.686],
       [28, 0, 1, 0, 19.796],
       [58, 0, 0, 0, 19.416],
       [67, 1, 2, 1, 10.898],
       [62, 1, 1, 1, 27.183],
       [24, 0, 0, 1, 18.457],
       [68, 0, 0, 1, 10.189],
       [26, 0, 1, 0, 14.16],
       [65, 1, 0, 1, 11.34],
```

```
from sklearn import preprocessing
from sklearn.model_selection  import train_test_split
```

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=4)
```

```
X_train.shape
```

```
(160, 5)
```

```
X_test.shape
```

```
(40, 5)
```

```
DecTree = DecisionTreeClassifier(criterion='entropy',max_depth=4)
DecTree
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

```
DecTree.fit(X_train,Y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

```
Y_hat=DecTree.predict(X_test)
Y_hat
```

```
array(['drugY', 'drugY', 'drugY', 'drugY', 'drugC', 'drugX', 'drugY',
       'drugY', 'drugY', 'drugA', 'drugA', 'drugX', 'drugA', 'drugY',
       'drugY', 'drugY', 'drugY', 'drugX', 'drugA', 'drugC', 'drugX',
       'drugC', 'drugA', 'drugX', 'drugC', 'drugB', 'drugX', 'drugY',
       'drugX', 'drugY', 'drugB', 'drugC', 'drugX', 'drugX', 'drugY',
       'drugY', 'drugA', 'drugA', 'drugX', 'drugY'], dtype=object)
```

```
from sklearn import metrics
metrics.accuracy_score(Y_hat,Y_test)
```

```
0.95
```

SVM (Cancer Data Set)

```
!wget -O cell_samples.csv https://cf-courses-data.s3.us.cloud-object-storage.appdomair
```

```
--2022-06-20 16:30:41--  https://cf-courses-data.s3.us.cloud-object-storage.appdomair
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-
HTTP request sent, awaiting response... 200 OK
Length: 19975 (20K) [text/csv]
Saving to: ‘cell_samples.csv’

cell_samples.csv    100%[===================>]  19.51K  --.-KB/s    in 0s

2022-06-20 16:30:41 (165 MB/s) - ‘cell_samples.csv’ saved [19975/19975]
```

```
import pandas as pd
```

```
df = pd.read_csv("cell_samples.csv")
df.head()
```

	ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	Nc
0	1000025	5	1	1	1	2	1	3	
1	1002945	5	4	4	5	7	10	3	
2	1015425	3	1	1	1	2	2	3	
3	1016277	6	8	8	1	3	4	3	
4	1017023	4	1	1	3	2	1	3	

```
df.columns
```

```
Index(['ID', 'Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize',
       'BareNuc', 'BlandChrom', 'NormNucl', 'Mit', 'Class'],
      dtype='object')
```

```
df=df[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize','BareNuc', 'BlandChr
```

```
df
```

	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl
0	5	1	1	1	2	1	3	1
1	5	4	4	5	7	10	3	2
2	3	1	1	1	2	2	3	1
3	6	8	8	1	3	4	3	7
4	4	1	1	3	2	1	3	1
...	...	...	...	...	...	...	...	...
694	3	1	1	1	3	2	1	1
695	2	1	1	1	2	1	1	1
696	5	10	10	3	7	3	8	10
697	4	8	6	4	3	4	10	6
698	4	8	8	5	4	5	10	4

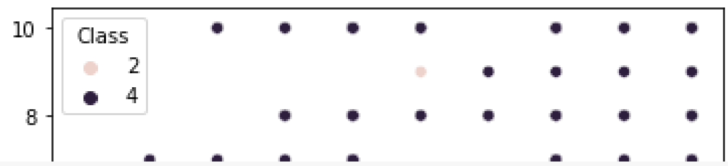
699 rows × 10 columns

```
df.columns
```

```
Index(['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize', 'BareNuc',
       'BlandChrom', 'NormNucl', 'Mit', 'Class'],
      dtype='object')
```

```
import seaborn as sns
sns.scatterplot(df.Clump,df.UnifSize,hue=df.Class)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas:
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f34823a3790>
```




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Clump        699 non-null    int64
 1   UnifSize     699 non-null    int64
 2   UnifShape    699 non-null    int64
 3   MargAdh      699 non-null    int64
 4   SingEpiSize  699 non-null    int64
 5   BareNuc      699 non-null    object
 6   BlandChrom   699 non-null    int64
 7   NormNucl     699 non-null    int64
 8   Mit          699 non-null    int64
 9   Class        699 non-null    int64
dtypes: int64(9), object(1)
memory usage: 54.7+ KB
```

```
df['BareNuc'] = df['BareNuc'].astype('int')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-f54d3735dc60> in <module>()
----> 1 df['BareNuc'] = df['BareNuc'].astype('int')

                              7 frames
/usr/local/lib/python3.7/dist-packages/pandas/_libs/lib.pyx in
pandas._libs.lib.astype_intsafe()

ValueError: invalid literal for int() with base 10: '?'
```

SEARCH STACK OVERFLOW

```
df = df[pd.to_numeric(df['BareNuc'], errors='coerce').notnull()]
df['BareNuc'] = df['BareNuc'].astype('int')
df.dtypes
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarnir
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/u:

Clump          int64
UnifSize       int64
UnifShape      int64
MargAdh        int64
```

```
SingEpiSize    int64
BareNuc        int64
BlandChrom     int64
NormNucl       int64
Mit            int64
Class          int64
dtype: object
```

```
df.head()
```

	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl
0	5	1	1	1	2	1	3	1
1	5	4	4	5	7	10	3	2
2	3	1	1	1	2	2	3	1
3	6	8	8	1	3	4	3	7
4	4	1	1	3	2	1	3	1

```
X=df.values[:,:-1]
X
```

```
array([[ 5,  1,  1, ...,  3,  1,  1],
       [ 5,  4,  4, ...,  3,  2,  1],
       [ 3,  1,  1, ...,  3,  1,  1],
       ...,
       [ 5, 10, 10, ...,  8, 10,  2],
       [ 4,  8,  6, ..., 10,  6,  1],
       [ 4,  8,  8, ..., 10,  4,  1]])
```

```
Y=df.values[:,-1]
Y
```

```
array([2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 4, 2, 4, 4, 2, 2, 4, 2, 4, 4,
       2, 2, 4, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 4, 2, 4, 4, 4, 4, 4, 4, 2,
       4, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 4, 2, 4, 2, 4,
       4, 2, 2, 4, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 2, 4, 2, 4,
       4, 4, 2, 2, 2, 4, 2, 2, 2, 2, 4, 4, 4, 2, 4, 2, 4, 2, 2, 2, 4, 2,
       2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 4, 2, 2, 4, 2, 4, 4, 2, 2, 4, 2, 2,
       4, 4, 2, 2, 2, 2, 4, 4, 2, 2, 2, 2, 2, 4, 4, 4, 2, 4, 2, 4, 2, 2,
       2, 4, 4, 2, 4, 4, 4, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 2, 2,
       2, 4, 4, 2, 2, 2, 4, 4, 2, 4, 4, 4, 2, 2, 4, 2, 2, 4, 4, 4, 4, 2,
       4, 4, 2, 4, 4, 4, 2, 4, 2, 4, 4, 4, 4, 2, 2, 2, 2, 2, 2, 4, 4, 2,
       2, 4, 2, 4, 4, 4, 2, 2, 2, 2, 4, 4, 4, 4, 4, 2, 4, 4, 4, 2, 4, 2,
       4, 4, 2, 2, 2, 2, 4, 2, 2, 4, 4, 4, 4, 4, 2, 4, 4, 2, 2, 4, 4, 2,
       2, 4, 4, 2, 4, 2, 4, 4, 2, 2, 4, 2, 2, 2, 4, 2, 2, 4, 4, 2, 2, 4,
       2, 4, 2, 2, 4, 2, 4, 4, 4, 2, 2, 4, 4, 2, 4, 2, 2, 4, 4, 2, 2, 2,
       4, 2, 2, 2, 4, 4, 2, 2, 2, 4, 2, 2, 4, 4, 4, 4, 4, 4, 2, 2, 2, 2,
       4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 4, 2,
       2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 4, 2, 4, 2, 4, 2, 2, 2, 2, 4, 2, 2, 2, 4, 2, 4, 2, 2, 2, 2, 2,
       2, 2, 4, 4, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 4, 2,
```

```
       4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 4, 2, 2, 4, 4, 2, 2, 2, 4, 4, 4, 2, 4, 2, 4, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 4, 4, 2, 2, 2, 4, 2,
       2, 4, 4, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       4, 2, 2, 4, 4, 4, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 4, 4, 2, 2, 2, 4,
       2, 4, 2, 4, 4, 4, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 2, 2, 4,
       2, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2,
       2, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 4, 4,
       4])
```

```
X.shape,Y.shape
```

```
((683, 9), (683,))
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=4)
```

SVM algorithm offers a choice of kernel functions for performing its processing.

mapping data into a higher dimensional space is called kernelling.

The mathematical function used for the transformation is known as the kernel function, and can be of different types, such as:

1.Linear 2.Polynomial 3.Radial basis function (RBF) 4.Sigmoid

```
from sklearn import svm
```

```
SVM = svm.SVC(kernel='rbf')
SVM.fit(X_train,Y_train)
```

```
SVC()
```

```
Y_hat=SVM.predict(X_test)
Y_hat
```

```
array([2, 4, 2, 4, 2, 2, 2, 2, 4, 2, 2, 4, 4, 4, 4, 2, 2, 2, 2, 2, 4, 2,
       4, 4, 4, 4, 2, 2, 4, 4, 4, 2, 4, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 4,
       4, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 4, 4, 2, 4, 4,
       4, 2, 2, 2, 4, 4, 2, 2, 2, 4, 2, 2, 4, 4, 2, 2, 2, 2, 4, 4, 2, 4,
       2, 2, 4, 4, 2, 2, 2, 4, 2, 2, 2, 4, 2, 4, 2, 2, 4, 2, 4, 2, 2, 4,
       2, 2, 4, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 2, 2, 4, 2, 2, 4, 2, 4, 2,
       2, 2, 2, 2, 4])
```

```
from sklearn import metrics
metrics.accuracy_score(Y_test,Y_hat)
```

```
0.9635036496350365
```

```
from sklearn.metrics import confusion_matrix
```

```
cnf_matrix = confusion_matrix(Y_test, Y_hat, labels=[2,4])
cnf_matrix
```

```
array([[85,  5],
       [ 0, 47]])
```

K MEANS clustering (Customer Segmentation)

```
!wget -O Cust_Segmentation.csv https://cf-courses-data.s3.us.cloud-object-storage.app
```

```
--2022-06-20 10:42:27--  https://cf-courses-data.s3.us.cloud-object-storage.appdomain
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-data
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-courses-
HTTP request sent, awaiting response... 200 OK
Length: 33426 (33K) [text/csv]
Saving to: ‘Cust_Segmentation.csv’

Cust_Segmentation.c 100%[===================>]  32.64K  --.-KB/s    in 0.02s

2022-06-20 10:42:27 (1.30 MB/s) - ‘Cust_Segmentation.csv’ saved [33426/33426]
```

```
df = pd.read_csv("Cust_Segmentation.csv")
df.head()
```

	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Defaulted	Address	DebtIncome
0	1	41	2	6	19	0.124	1.073	0.0	NBA001	
1	2	47	1	26	100	4.582	8.218	0.0	NBA021	
2	3	33	2	10	57	6.111	5.802	1.0	NBA013	
3	4	29	2	4	19	0.681	0.516	0.0	NBA009	

As you can see, Address in this dataset is a categorical variable. The k-means algorithm isn't directly applicable to categorical variables because the Euclidean distance function isn't really meaningful for discrete variables. So, let's drop this feature and run clustering

```
df=df.drop('Address',axis=1)
df.head()
```

	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Defaulted	DebtIncomeRatio
0	1	41	2	6	19	0.124	1.073	0.0	6.3
1	2	47	1	26	100	4.582	8.218	0.0	12.8
2	3	33	2	10	57	6.111	5.802	1.0	20.9

```
from sklearn import preprocessing
```

```
from sklearn.preprocessing import StandardScaler
```

```
X = df.values[:,1:] #removing id column
X = np.nan_to_num(X)
Clus_dataSet = StandardScaler().fit_transform(X)
Clus_dataSet
```

```
array([[ 0.74291541,  0.31212243, -0.37878978, ..., -0.59048916,
        -0.52379654, -0.57652509],
       [ 1.48949049, -0.76634938,  2.5737211 , ...,  1.51296181,
        -0.52379654,  0.39138677],
       [-0.25251804,  0.31212243,  0.2117124 , ...,  0.80170393,
         1.90913822,  1.59755385],
       ...,
       [-1.24795149,  2.46906604, -1.26454304, ...,  0.03863257,
         1.90913822,  3.45892281],
       [-0.37694723, -0.76634938,  0.50696349, ..., -0.70147601,
        -0.52379654, -1.08281745],
       [ 2.1116364 , -0.76634938,  1.09746566, ...,  0.16463355,
        -0.52379654, -0.2340332 ]])
```

```
from sklearn.cluster import KMeans
```

```
k = 3
k_means = KMeans(init = "k-means++", n_clusters = k, n_init = 12)
k_means.fit(X)
```

```
KMeans(n_clusters=3, n_init=12)
```

```
labels = k_means.labels_
```

```
labels
```

```
array([1, 2, 1, 1, 0, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1,
       1, 2, 2, 2, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 0,
       1, 2, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1, 2, 1, 2, 1, 2, 2, 1, 1,
       2, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 2, 1, 2, 2, 0, 1, 1, 1, 1, 1,
       1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 2,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2,
       1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1,
       2, 1, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
       1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 2,
       0, 1, 2, 1, 1, 1, 1, 1, 1, 0, 2, 1, 1, 1, 1, 2, 1, 1, 2, 2, 1, 2,
```

```
       1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1,
       2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 1,
       2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 2, 1,
       2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 2, 1,
       1, 1, 1, 1, 2, 1, 1, 1, 0, 1, 1, 1, 2, 1, 2, 2, 2, 1, 1, 1, 2, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 2, 1,
       1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 0, 1, 1, 1, 1, 2, 1, 0,
       1, 1, 1, 1, 2, 1, 2, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1,
       1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 1, 1,
       1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 2, 1,
       1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1,
       1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 1, 2, 1, 1, 2,
       1, 2, 1, 1, 0, 1, 2, 1, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 1,
       1, 1, 2, 2, 1, 1, 2, 1, 1, 1, 2, 1, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 1, 1, 2, 1, 2, 1, 2, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 2,
       1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 2,
       1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 0,
       2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2], dtype=int32)
```

```
1 df.head()
```

	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Defaulted	DebtIncomeRatio
0	1	41	2	6	19	0.124	1.073	0.0	6.3
1	2	47	1	26	100	4.582	8.218	0.0	12.8
2	3	33	2	10	57	6.111	5.802	1.0	20.9
3	4	29	2	4	19	0.681	0.516	0.0	6.3
4	5	47	1	31	253	9.308	8.908	0.0	7.2

```
1 df['Clus_km']=labels
2 df.head()
```

```
1 df.groupby('Clus_km').mean()
```

Clus_km	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	D
0	410.166667	45.388889	2.666667	19.555556	227.166667	5.678444	10.907167	
1	432.468413	32.964561	1.614792	6.374422	31.164869	1.032541	2.104133	

```
1 df.groupby('Clus_km').mean().corr()
```

	Customer Id	Age	Edu	Years Employed	Income	Card Debt	Oth De
Customer Id	1.000000	-0.836471	-0.559038	-0.836649	-0.507732	-0.667231	-0.6412
Age	-0.836471	1.000000	0.921998	1.000000	0.896823	0.966306	0.9569
Edu	-0.559038	0.921998	1.000000	0.921872	0.998160	0.990595	0.9947
Years Employed	-0.836649	1.000000	0.921872	1.000000	0.896679	0.966222	0.9568
Income	-0.507732	0.896823	0.998160	0.896679	1.000000	0.980475	0.9866
Card Debt	-0.667231	0.966306	0.990595	0.966222	0.980475	1.000000	0.9994
Other Debt	-0.641298	0.956912	0.994704	0.956817	0.986641	0.999412	1.0000
Defaulted	0.698847	-0.192589	0.202379	-0.192908	0.261389	0.066477	0.1006

```
1 sns.swarmplot(x=df.Age,y=df.Income,hue=df.Clus_km)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 50.0
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 66.7
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 78.6
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 81.6
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 86.7
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 80.6
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 81.8
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 86.8
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 82.4
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 88.1
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 70.6
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 83.9
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 81.6
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 82.5
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 72.7
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 71.6
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 75.6
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 59.4
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 61.1
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 45.5
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 33.3
  warnings.warn(msg, UserWarning)
```

```
sns.swarmplot(x=df.Edu,y=df.Income,hue=df.Clus_km)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 77.6
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 59.1
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 17.8
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 26.5
  warnings.warn(msg, UserWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7ff420a2c9d0>
```