set up the envirement:

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
from sklearn.feature_selection import mutual_info_regression
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestRegressor
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import cross_val_score
import pickle
#from tqdm import notebook
warnings.filterwarnings('ignore')
plt.style.use("seaborn-whitegrid")
plt.rc("figure", autolayout=True)
plt.rc(
    "axes",
    labelweight="bold",
    labelsize="large",
    titleweight="bold",
    titlesize=14,
    titlepad=10,
)
print('set up complited')
```

```
set up complited
```

useful function:

In [ ]:

```
def make_MI_scores(X,y) :
    mi_scores = mutual_info_regression(X,y)
    mi_scores = pd.Series(mi_scores,index=X.columns)
    mi_scores = mi_scores.sort_values(ascending=False)

    return mi_scores
```

load the data:

In [ ]:

```
data_DS = pd.read_csv('./data_DS.csv',index_col=0)
data_PEC = pd.read_csv('./data_PEC.csv',index_col=0)
```

working with DS

In [ ]:

```
data_DS.head()
```

Out[ ]:

	Genre	ville	CJT	Enf	Mt_remb	Age	ALD
0	Masculin	AGADIR	1	2	4040.00	61	1.0
1	Masculin	CASABLANCA	1	2	3150.56	57	1.0
2	Masculin	CASABLANCA	1	2	3150.56	58	1.0
3	Masculin	CASABLANCA	1	2	31191.20	62	2.0
4	Masculin	CASABLANCA	1	2	31191.20	63	2.0

In [ ]:

```
data_DS.isnull().sum()
```

Out[ ]:

```
Genre         0
ville         0
CJT           0
Enf           0
Mt_remb       0
Age           0
ALD        1341
dtype: int64
```

In [ ]:

```
X = data_DS.copy()
y = X.pop('Mt_remb')
X['Genre'],_ = X.Genre.factorize()
X['ville'],_ = X.ville.factorize()
X.fillna(0,inplace=True)
```

In [ ]:

```
mutual_info_regression(X,y)
```

Out[ ]:

```
array([0.16779877, 0.89462879, 0.09305668, 0.85048718, 0.65060166,
       0.75631101])
```

In [ ]:

```
mi_scores = make_MI_scores(X,y)
```

In [ ]:

```
mi_df = pd.DataFrame({'score':mi_scores,'col':mi_scores.index})
```

In [ ]:

```
mi_df
```

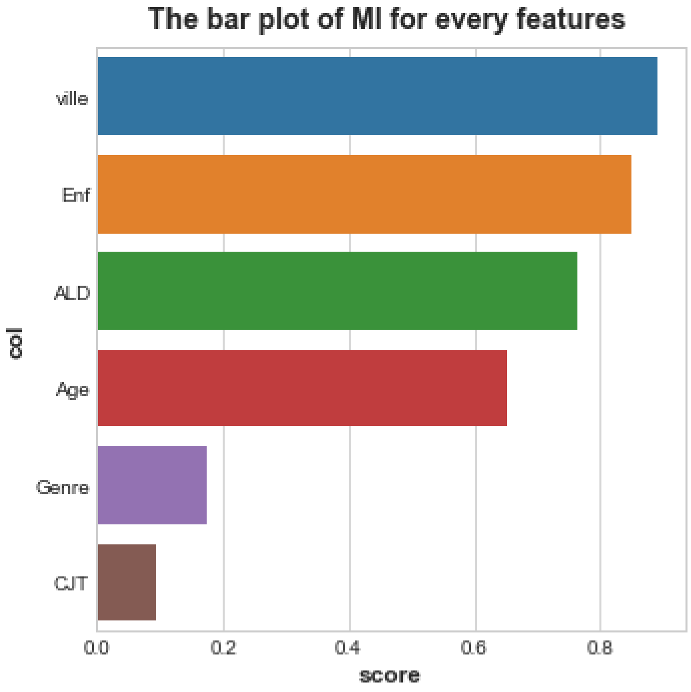
Out[ ]:

	score	col
ville	0.889964	ville
Enf	0.849347	Enf
ALD	0.763821	ALD
Age	0.652215	Age
Genre	0.173301	Genre
CJT	0.093797	CJT

In [ ]:

```
plt.figure(figsize=(5,5))
plt.title('The bar plot of MI for every features ')

sns.barplot(x='score',y='col',data=mi_df)
plt.show()
```



create new features:

In [ ]:

```
X = data_DS.copy()
y = X.pop('Mt_remb')
```

In [ ]:

```
X['fammilySize'] = X['CJT']+X['Enf']+1
X['Genre'],_ = X.Genre.factorize()
X['ville'],_ = X.ville.factorize()
```

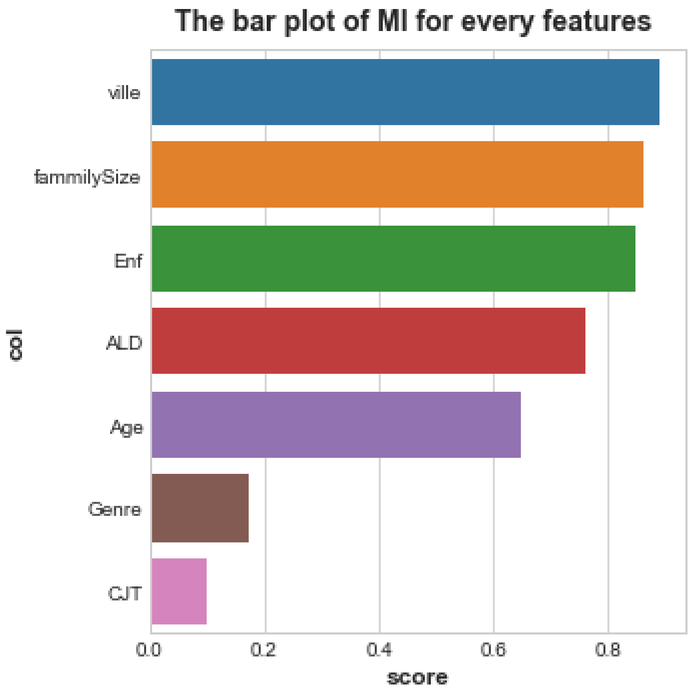
In [ ]:

```
X.fillna(0,inplace=True)
mi_scores = make_MI_scores(X,y)
```

In [ ]:

```
mi_df = pd.DataFrame({'score':mi_scores,'col':mi_scores.index})
plt.figure(figsize=(5,5))
plt.title('The bar plot of MI for every features ')

sns.barplot(x='score',y='col',data=mi_df)
plt.show()
```



In [ ]:

```
X['new_ville'] = data_DS.ville.map(lambda ville: ville if ville in ['CASABLANCA', 'MARR
AKECH', 'MOHAMMEDIA', 'AGADIR', 'OUJDA', 'TANGER',
       'KENITRA', 'SETTAT', 'ELJADIDA', 'FES', 'MEKNES', 'RABAT', 'TEMARA'] else 'OTHE
R')
```

In [ ]:

```
X.new_ville.value_counts()
```

Out[ ]:

```
CASABLANCA    12780
OTHER          2324
MARRAKECH       919
MOHAMMEDIA      765
AGADIR          710
OUJDA           535
TANGER          510
KENITRA         488
SETTAT          474
ELJADIDA        421
FES             365
MEKNES          325
RABAT           320
TEMARA          228
Name: new_ville, dtype: int64
```

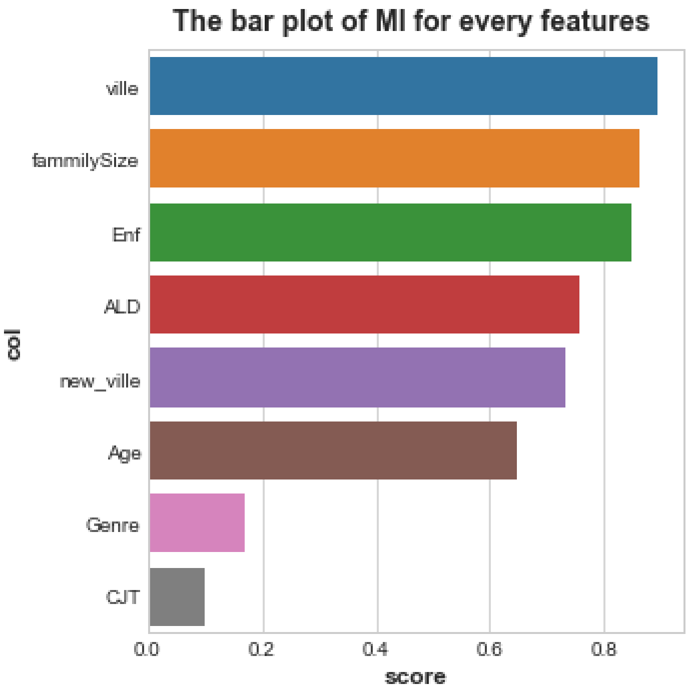
In [ ]:

```
X['new_ville'],_ = X['new_ville'].factorize()
```

In [ ]:

```
X.fillna(0,inplace=True)
mi_scores = make_MI_scores(X,y)
mi_df = pd.DataFrame({'score':mi_scores,'col':mi_scores.index})
plt.figure(figsize=(5,5))
plt.title('The bar plot of MI for every features ')

sns.barplot(x='score',y='col',data=mi_df)
plt.show()
```



In [ ]:

```
class Preprocessing:
   def __init__(self) -> None:
      self.cluster = None
      self.trained = False
   def add_features(self,df,target) -> tuple:
      X = df.copy()
      y = X.pop(target)
      X['ville'] = X.ville.map(lambda ville: ville if ville in ['CASABLANCA', 'MARRAKECH', 'MOHAMMEDIA', 'AGADIR', 'OUJDA', 'TANGER',
       'KENITRA', 'SETTAT', 'ELJADIDA', 'FES', 'MEKNES', 'RABAT', 'TEMARA'] else 'OTHER')
      X['fammilySize'] = X['CJT']+X['Enf']+1
      return X,y
   def fit(self,X_cluster) -> None:
      for col in X_cluster.select_dtypes('object').columns:
         X_cluster[col],_ = X_cluster[col].factorize()
         X_cluster.fillna(0,inplace=True)
      self.cluster = KMeans(n_clusters=10,random_state=0)
      self.cluster.fit(X_cluster)
   def predict(self,X_cluster):
      for col in X_cluster.select_dtypes('object').columns:
         X_cluster[col],_ = X_cluster[col].factorize()
         X_cluster.fillna(0,inplace=True)
      return self.cluster.predict(X_cluster)

   def transform(self,df,target:str) ->tuple:
      X,y = self.add_features(df,target)
      if not self.trained:
         self.fit(X.copy())
      self.trained = True
      X['cluster'] = self.predict(X.copy())
      return X,y
```

In [ ]:

```
for i in range(3,10):
    my_cluster = KMeans(n_clusters=i,random_state=0)
    X_cluster = X.copy()
    X_cluster['cluster'] = my_cluster.fit_predict(X_cluster)
    score = make_MI_scores(X_cluster,y)
    print(i,'-->',score['cluster'])
```

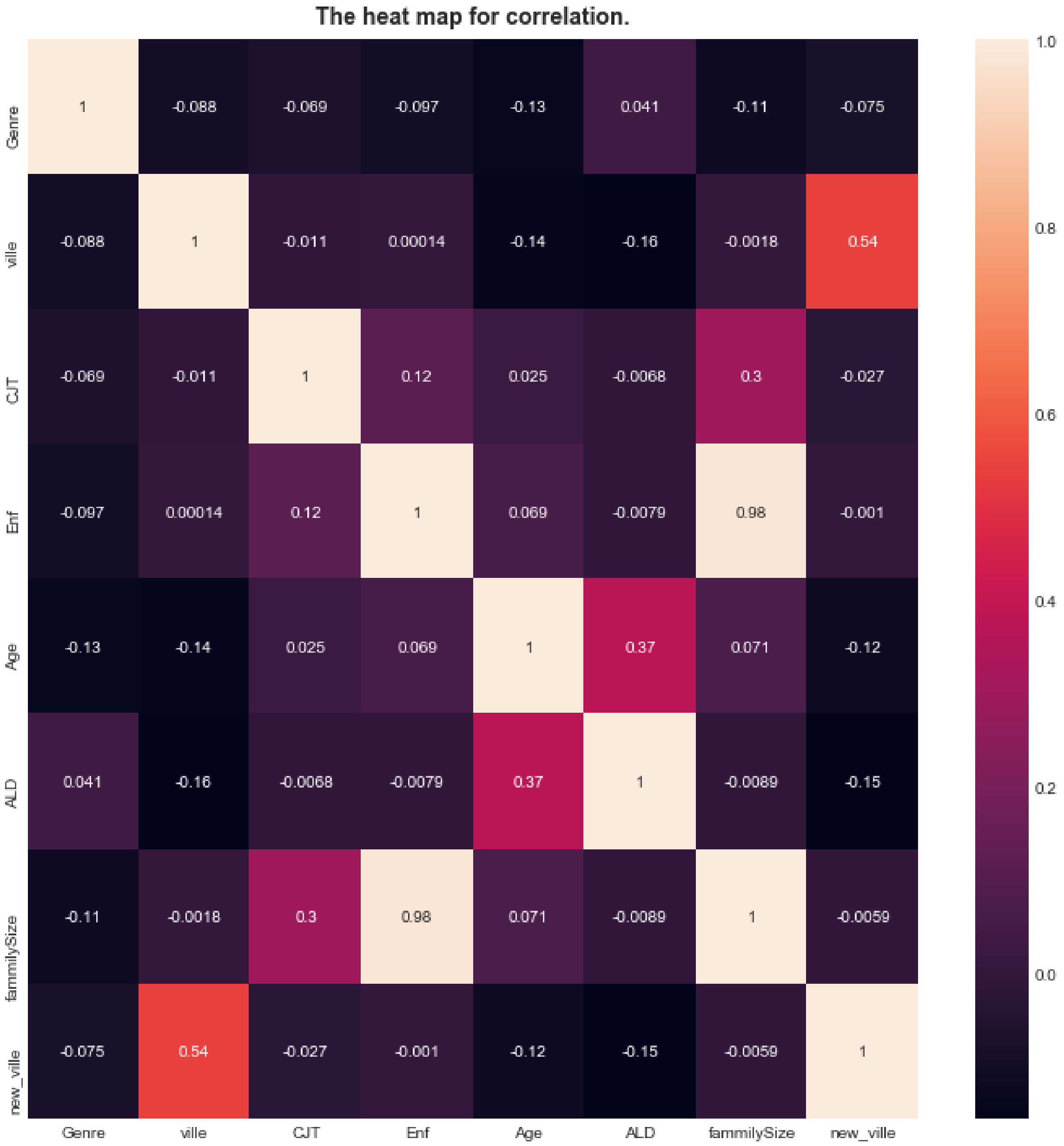
```
3 --> 0.37840318765587533
4 --> 0.5987722398872204
5 --> 0.727870818280377
6 --> 0.7503507256391071
7 --> 0.866322743357661
8 --> 0.8932031460406722
9 --> 0.9171242525085135
```

In [ ]:

```
my_cluster = KMeans(n_clusters=10,random_state=0)
X_cluster = X.copy()
_ =my_cluster.fit(X_cluster)
```

In [ ]:

```
plt.figure(figsize=(10,10))
plt.title('The heat map for correlation.')
sns.heatmap(X.corr(),annot=True)
plt.show()
```



parameter tuning:

In [ ]:

```
my_preproccing = Preprocessing()
X_full,y_full = my_preproccing.transform(data_DS,'Mt_remb')
```

In [ ]:

```
numerical_cols = X_full.select_dtypes(['float','int']).columns
categorical_cols = X_full.select_dtypes(['object']).columns
```

In [ ]:

```
# Preprocessing for numerical data
numerical_transformer = SimpleImputer(strategy='median')

# Preprocessing for categorical data
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore',sparse=False))
])

# Bundle preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cateee', categorical_transformer, categorical_cols)
    ])
```

In [ ]:

```
for i in range(100,1000,50):
    model = RandomForestRegressor(n_estimators=i, random_state=0)
    My_Pipe_Line = Pipeline(steps=[
        ('preprocessor',preprocessor),
        ('model',model)
    ])
    error = -1 *cross_val_score(My_Pipe_Line,X_full,y_full,cv=5,scoring='neg_mean_absol
ute_error').mean()
    print(i,'--->',error)
```

```
100 ---> 13622.522149165201
150 ---> 13619.616716107901
200 ---> 13615.185554454089
250 ---> 13614.00136698635
300 ---> 13616.437105209
350 ---> 13615.263628674405
400 ---> 13613.418735459543
450 ---> 13611.513157630605
500 ---> 13613.256059617104
550 ---> 13614.2270301633
600 ---> 13614.542097510572
650 ---> 13613.968418781811
700 ---> 13614.681891577397
750 ---> 13615.231067500286
800 ---> 13616.496437434564
850 ---> 13615.89382164757
900 ---> 13615.792281456448
950 ---> 13615.734089185315
```

In [ ]:

```
for i in range(400,500,10):
    model = RandomForestRegressor(n_estimators=i, random_state=0)
    My_Pipe_Line = Pipeline(steps=[
        ('preprocessor',preprocessor),
        ('model',model)
    ])
    error = -1 *cross_val_score(My_Pipe_Line,X_full,y_full,cv=5,scoring='neg_mean_absol
ute_error').mean()
    print(i,'--->',error)
```

```
400 ---> 13613.418735459543
410 ---> 13612.755135504136
420 ---> 13611.650516409281
430 ---> 13611.776724544883
440 ---> 13612.1743364416
450 ---> 13611.513157630605
460 ---> 13611.545536120824
470 ---> 13611.40533156286
480 ---> 13612.99233985277
490 ---> 13613.4003500083
```

In [ ]:

```
for i in range(460,480,5):
    model = RandomForestRegressor(n_estimators=i, random_state=0)
    My_Pipe_Line = Pipeline(steps=[
        ('preprocessor',preprocessor),
        ('model',model)
    ])
    error = -1 *cross_val_score(My_Pipe_Line,X_full,y_full,cv=5,scoring='neg_mean_absol
ute_error').mean()
    print(i,'--->',error)
```

```
460 ---> 13611.545536120824
465 ---> 13611.678519123618
470 ---> 13611.40533156286
475 ---> 13612.679583380406
```

training using X_full and y_full:

In [ ]:

```
model = RandomForestRegressor(n_estimators=470, random_state=0)
My_Pipe_Line_DS = Pipeline(steps=[
    ('preprocessor',preprocessor),
    ('model',model)
])
My_Pipe_Line_DS.fit(X_full,y_full)
```

Out[ ]:

```
Pipeline(memory=None,
     steps=[('preprocessor', ColumnTransformer(n_jobs=None, remainder='dro
p', sparse_threshold=0.3,
         transformer_weights=None,
         transformers=[('num', SimpleImputer(copy=True, fill_value=None, m
issing_values=nan,
       strategy='median', verbose=0), Index(['ALD', 'cluster'], dtype='obj
ect...imators=470, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=Fals
e))])
```

save the model so that we can load it in other project:

In [ ]:

```
filename = 'finalized_model_for_DS.nav'
pickle.dump(My_Pipe_Line,open(filename,'wb'))
```

model for PEC:

In [ ]:

```
data_PEC = pd.read_csv('./data_PEC.csv',index_col=0)
```

In [ ]:

```
data_PEC.head()
```

Out[ ]:

	Matricule	CJT	Enf	ALD	MT_PEC	Age	Genre
0	10016	1	2	1	141141.196	63.0	Masculin
5	10018	1	2	1	891492.240	60.0	Masculin
11	1004	1	2	2	115206.916	63.0	Masculin
17	10046	1	3	1	5859.675	NaN	NaN
18	10047	1	3	0	31254.400	66.0	Masculin

In [ ]:

```
X_full,y_full = my_preproccing.transform(data_DS,'Mt_remb')
```

In [ ]:

```
X_full.head()
```

Out[ ]:

	Genre	ville	CJT	Enf	Age	ALD	fammilySize	cluster
0	Masculin	AGADIR	1	2	61	1.0	4	0
1	Masculin	CASABLANCA	1	2	57	1.0	4	4
2	Masculin	CASABLANCA	1	2	58	1.0	4	4
3	Masculin	CASABLANCA	1	2	62	2.0	4	0
4	Masculin	CASABLANCA	1	2	63	2.0	4	0

In [ ]:

```
numerical_cols = X_full.select_dtypes(['float','int']).columns
categorical_cols = X_full.select_dtypes(['object']).columns
```

In [ ]:

```
# Preprocessing for numerical data
numerical_transformer = SimpleImputer(strategy='median')

# Preprocessing for categorical data
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore',sparse=False))
])

# Bundle preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cateee', categorical_transformer, categorical_cols)
    ])
```

In [ ]:

```
for i in range(100,1000,50):
    model = RandomForestRegressor(n_estimators=i, random_state=0)
    My_Pipe_Line_PEC = Pipeline(steps=[
        ('preprocessor',preprocessor),
        ('model',model)
    ])
    error = -1 *cross_val_score(My_Pipe_Line_PEC,X_full,y_full,cv=5,scoring='neg_mean_absolute_error').mean()
    print(i,'--->',error)
```

```
100 ---> 13622.522149165201
150 ---> 13619.616716107901
200 ---> 13615.185554454089
250 ---> 13614.00136698635
300 ---> 13616.437105209
350 ---> 13615.263628674405
400 ---> 13613.418735459543
450 ---> 13611.513157630605
500 ---> 13613.256059617104
550 ---> 13614.2270301633
600 ---> 13614.542097510572
650 ---> 13613.968418781811
700 ---> 13614.681891577397
750 ---> 13615.231067500286
800 ---> 13616.496437434564
850 ---> 13615.89382164757
900 ---> 13615.792281456448
950 ---> 13615.734089185315
```

In [ ]:

```
for i in range(400,500,5):
    model = RandomForestRegressor(n_estimators=i, random_state=0)
    My_Pipe_Line_PEC = Pipeline(steps=[
        ('preprocessor',preprocessor),
        ('model',model)
    ])
    error = -1 *cross_val_score(My_Pipe_Line_PEC,X_full,y_full,cv=5,scoring='neg_mean_a
bsolute_error').mean()
    print(i,'--->',error)
```

```
400 ---> 13613.418735459543
405 ---> 13613.043192887952
410 ---> 13612.755135504136
415 ---> 13612.34875651006
420 ---> 13611.650516409281
425 ---> 13611.66739208451
430 ---> 13611.776724544883
435 ---> 13611.768358703102
440 ---> 13612.1743364416
445 ---> 13611.84798208827
450 ---> 13611.513157630605
455 ---> 13611.43198255773
460 ---> 13611.545536120824
465 ---> 13611.678519123618
470 ---> 13611.40533156286
475 ---> 13612.679583380406
480 ---> 13612.99233985277
485 ---> 13613.030242935649
490 ---> 13613.4003500083
495 ---> 13613.505130964146
```

In [ ]:

```
model = RandomForestRegressor(n_estimators=470, random_state=0)
My_Pipe_Line_PEC = Pipeline(steps=[
    ('preprocessor',preprocessor),
    ('model',model)
])
My_Pipe_Line_PEC.fit(X_full,y_full)
```

Out[ ]:

```
Pipeline(memory=None,
     steps=[('preprocessor', ColumnTransformer(n_jobs=None, remainder='dro
p', sparse_threshold=0.3,
         transformer_weights=None,
         transformers=[('num', SimpleImputer(copy=True, fill_value=None, m
issing_values=nan,
       strategy='median', verbose=0), Index(['ALD', 'cluster'], dtype='obj
ect...imators=470, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=Fals
e))])
```

In [ ]:

```
filename = 'finalized_model_for_PEC.nav'
pickle.dump(My_Pipe_Line,open(filename,'wb'))
```

In [ ]:

```
filename = 'cluster.nav'
pickle.dump(my_cluster,open(filename,'wb'))
```

In [ ]:

```
### to load the model we use this line of code
model = pickle.load(open(filename,'rb'))
```

the end