My Decision For Model:

For my very first Submission I have used a simple **Decision Tree Regressor Model** , after seeing the accuracy with this model's result , I will add on other features(including XGBoost etc) to improve upon the results.

Major Steps and my Philosophy Involved:

a) Exploring Dataset : Getting to know the data

For my references : Shape,description , info , If any null values, name of columns , dtype of columns etc

b) Dealing with Null Values:

There are many null values in the training data set : Out of 50882 -- 27334 rows have got null values , deleting all these rows will be a huge loss of data to us.

As many fields contain **categorical values** , I have decided to **replace null values with most frequently occurring attributes** in respective columns using **Mode function**

c) Processing Data - Getting Data Ready for Model:

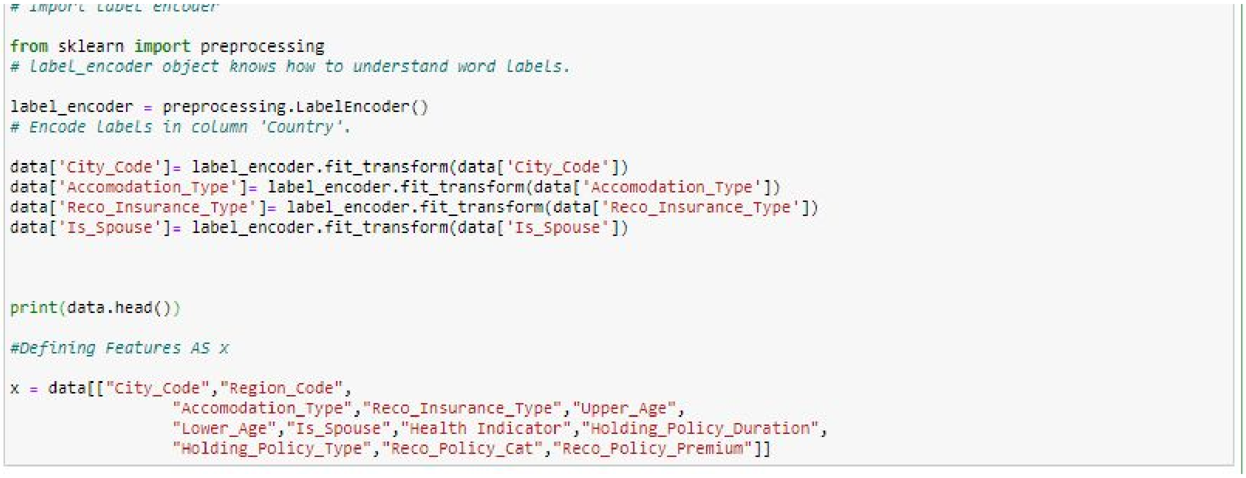
I thought to use the **One-Hot-Encodin**g , but before going ahead with that, there are 2 columns that are neither string nor float/int , thus i need to process these columns differently.

For these 2 columns , **Health Indicator and Holding_Policy_Duration, I** specifically replaced all the unique attributes in the column using a corresponding int value, so as to categorise.

For Example: For Health Indicator Column,
data['Health Indicator'].replace
({"X1":1,"X2":2,"X3":3,"X4":4,"X5":5,"X6":6,"X7":7,"X8":8,"X9":9},inplace=True)

Processing Other Columns with One-Hot-Encoder:

I have processed the remaining columns by using sklearn Label Encoder.

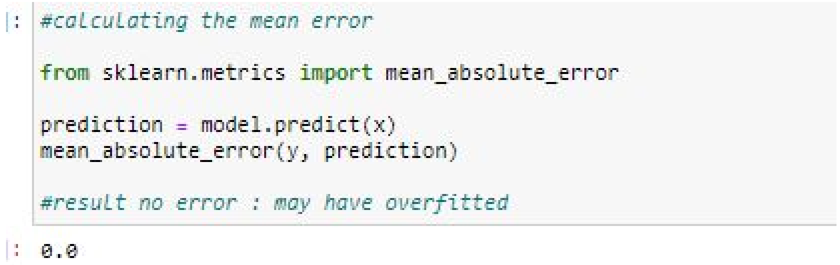


```
# import label encoder

from sklearn import preprocessing
# label_encoder object knows how to understand word labels.

label_encoder = preprocessing.LabelEncoder()
# Encode labels in column 'Country'.

data['City_Code']= label_encoder.fit_transform(data['City_Code'])
data['Accomodation_Type']= label_encoder.fit_transform(data['Accomodation_Type'])
data['Reco_Insurance_Type']= label_encoder.fit_transform(data['Reco_Insurance_Type'])
data['Is_Spouse']= label_encoder.fit_transform(data['Is_Spouse'])



print(data.head())

#Defining Features AS x

x = data[["City_Code","Region_Code",
                "Accomodation_Type","Reco_Insurance_Type","Upper_Age",
                "Lower_Age","Is_Spouse","Health Indicator","Holding_Policy_Duration",
                "Holding_Policy_Type","Reco_Policy_Cat","Reco_Policy_Premium"]]
```

d) **Building the model:**
As already mentioned , I used the SKLEARN decision tree regressor as a model for prediction.

e) Next , Obvious steps , Fitting the training data into the model , and making predictions out of it.For now , I have not splitted my training data into test/train, I will first see the result with this, and then make changes if required. **Our results can be overfitted and god knows what!**

I got 0 mean value error : Inclining Towards overfitting !



```
#calculating the mean error

from sklearn.metrics import mean_absolute_error

prediction = model.predict(x)
mean_absolute_error(y, prediction)

#result no error : may have overfitted
```

```
0.0
```

0--- DANGER!

f) **Repeating the same processing steps for Test-data**

Let's Hope for Good Results!