NAN の処理

1. 落とす。　data{" "].drop("data_columns",axis=1)

Data.dropna(subset=”saleplece”,inplace=True)

##subset=””である説明変数から見て欠損値があったとき、その行を消す。

また、inpace=Ture とすると、元のオブジェクト自体（data）が変更される。新しいものを作る必要がない。

また、axis=1 とすると、その列をすべて消す.

X_full.drop(['SalePrice'], axis=1, inplace=True)

みたいに使う

2　平均値の挿入

---

from sklearn.impute from SimpleImputer

Imputer_kodai=SimpleImputer(strategy="median")

final_X_train=pd.DataFrame(Imputer_kodai.fit_transform(X_rain)

final_X_valid=pd.DataFrame(imputer_kodai.transform(X_valid)

それか

---

final_X_train=X_train.fillna(np.mean(X_train["AAA"],insplace=True)

3.新しく要素を作る

カテゴリ変数の作成

以下のようなカテゴリ変数の処理について学ぶ

まず、文字列を含む項を以下のようにして割り出す

説明変数の中にデータ型（dtype）がテキストデータ（object）のものがあるのかを確認するプログラム

```
s=(X_train.dtypes==”object”)　　！　. dtypes はデータの型を指定できる.
                                  ！　 dtypes==”object”でデータ型が文字のものを指定
object_cols=list(s[s].index)
print(object_cols)
```

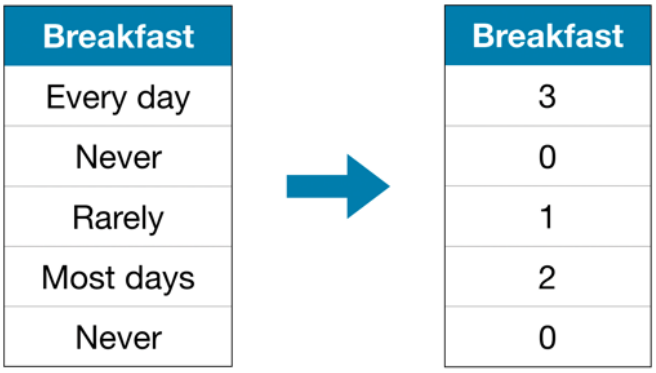
以下は処理の方法

1 [落とす]

単純に上と同様に消す

```
        Drop_X_train=X_train.select_dtypes(exclude=[“object”])
        !!　.select_dtypes()で型を指定.
!!　exclude=[“ “]で除外する型を指定できる
```

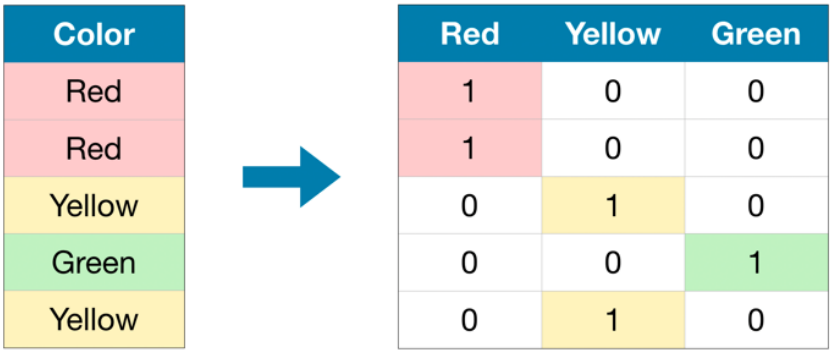
２ラベルエンコーディング



From sklearn.preprocessing import LabelEncoder
Label_X_train=X_train.copy()
Label_X_valid=X_valid.copy()
Label_kodai_concoder=LabeEncoder()
For col in object_cols:
　　Label_X_train[col]=label_kodai_encoder.fit_transform(X_train[col])
　　Label_X_valid[col]=label_kodai_encoder.transform(X_valid[col])

で、ラベルエンコーディングされた説明変数軍が作れる。

3 One-Hot-encoding



Skit-learn からまた持ってくる
この時、handle_unknown=”ignore”d でトレーニングデータで代表されるエラー（値が有効でないもの）をはじいてくれる
また、sparese=False では帰ってくる値を numpy の array で返すようにしている.

---
from sklearn.preprocessing import OneHotEncoder
#カテゴリかるデータを用いてワンホットエンコーディングを適用する
＃OH_encoder は他の説明変数は入っておらず、本当にワンホットエンコーディングだけのもの
OH_encoder = OneHotEncoder(handle_unknown=”ignore”,sparse=False)
OH_cols_train=pd.DataFrame(OH_encoder.fit_transform(X_train[object_cols]))
OH_cols_valid=pd.DataFrame(OH_encoder.fit_transform(X_valid[object_cols]))
＃ワンホットエンコーディングからインデックスを排除しているらしいけどよくわからん

OH_cols_train.index = X_train.index
OH_cols_train.index = X_train.index　??????????????
＃カテゴリかる列を排除（もうワンホットエンコーディングは行っているから）
Num_X_train=X_train.drop(object_cols,axis=1)
Num_X_valid=X_valid.drop(object_cols,axis=1)
#dtypes=”object“のものを排除した Num～とそれをワンホットエンコーディングした OH_~を結合（concat）する
OH_X_train=pd.concat([num_x_train,OH_cols_train],axis=1)
OH_X_valid=pd.concat([num_x_valid,OH_cols_valid],axis=1)

以下便利な技の説明
以下を使用すれば、値が文字列、かつ、それらが何個のカテゴリデータ数を持つかを返すことができる
# Get number of unique entries in each column with categorical data
object_nunique = list(map(lambda col: X_train[col].nunique(), object_cols))
d = dict(zip(object_cols, object_nunique))
＃↑をやると、ユニークな値（文字列）を持つ col の名前とそのカテゴリ変数の数を出力できる

# Print number of unique entries by column, in ascending order
sorted(d.items(), key=lambda x: x[1])

↑↑↑↑↑↑↑↑↑↑↑↑
A=[1,2,3,4,5,6]
List(map(lambda x:x^2 , A))　　map 関数は括弧内の動作を全体に適用するもの
　　x ^ 2 を A の要素分行う
　　list 関数はこれらをリスト化する

・set()関数　：重複をゆるさず、すべての値をリストで返す