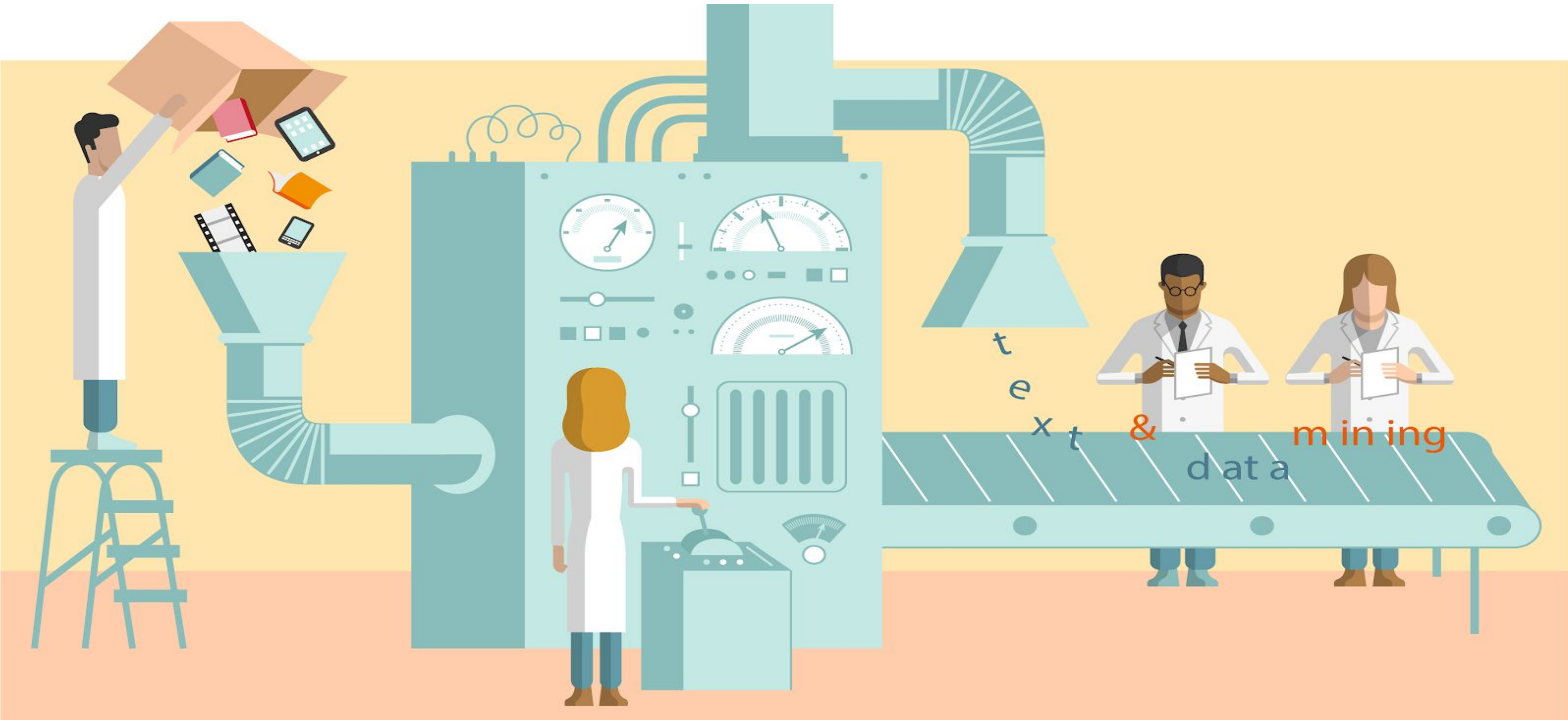
Assignment 4

Data Mining

Harisai - 17pw13

Data Imputation



Why data imputation ?

Missing data is a common problem in practical data analysis. In datasets, missing values could be represented as '?', 'nan', 'N/A', blank cell, or sometimes '-999', 'inf', '-inf'.

Imputation procedures in which the missing data values are replaced with some value is another commonly used strategy for dealing with missing value. These procedures result in a hypothetical 'complete' data set that will cause no problems with the analysis.

Data:-

The data consists of 43 column and 198900 rows with both categorical an numerical data.

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer

permits = pd.read_csv("Building_Permits.csv")

np.random.seed(0)
permits.head(10)
```

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Number Suffix	Street Name	Street Suffix	...	Existing Construction Type	Existing Construction Type Description	Proposed Construction Type	Proposed Construction Type Description	Site Permit	Supervisor District	Neighborhoods - Analysis Boundaries	Zipcode	Locat
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	NaN	Ellis	St	...	3.0	constr type 3	NaN	NaN	NaN	3.0	Tenderloin	94102.0	(37.7857192566807 -122.408523131948
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	NaN	Geary	St	...	3.0	constr type 3	NaN	NaN	NaN	3.0	Tenderloin	94102.0	(37.787339806007 -122.410631997577
2	201605278609	3	additions alterations or repairs	05/27/2016	0595	203	1647	NaN	Pacific	Av	...	1.0	constr type 1	1.0	constr type 1	NaN	3.0	Russian Hill	94109.0	(37.79465733242 -122.422325629792
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	NaN	Pacific	Av	...	5.0	wood frame (5)	5.0	wood frame (5)	NaN	3.0	Nob Hill	94109.0	(37.795958679091 -122.415574055194
4	201611283529	6	demolitions	11/28/2016	0342	001	950	NaN	Market	St	...	3.0	constr type 3	NaN	NaN	NaN	6.0	Tenderloin	94102.0	(37.783152618973 -122.409508839977
5	201706149344	8	otc alterations permit	06/14/2017	4105	009	800	NaN	Indiana	St	...	1.0	constr type 1	1.0	constr type 1	NaN	10.0	Potrero Hill	94107.0	(37.759223313465 -122.391704026285
6	201706300814	8	otc alterations permit	06/30/2017	1739	020	1291	NaN	11th	Av	...	5.0	wood frame (5)	5.0	wood frame (5)	NaN	5.0	Inner Sunset	94122.0	(37.7641456401385 -122.468751124703
7	M803667	8	otc alterations permit	06/30/2017	4789	014	1465	NaN	Revere	Av	...	NaN	NaN	NaN	NaN	NaN	10.0	Bayview Hunters Point	94124.0	(37.730050990236 -122.387849389166
8	M804227	8	otc alterations permit	07/05/2017	1212	054	2094	NaN	Fell	St	...	NaN	NaN	NaN	NaN	NaN	5.0	Lone Mountain/USF	94117.0	(37.7723934985025 -122.452314668246
9	M804767	8	otc alterations permit	07/06/2017	1259	016	89	NaN	Alpine	Tr	...	NaN	NaN	NaN	NaN	NaN	8.0	Haight Ashbury	94117.0	(37.76917242937 -122.437348590519

10 rows × 43 columns

The dataset contains the city permit data for various building operations.

Each city or county has its own office related to buildings, that can do multiple functions like issuing permits, inspecting buildings to enforce safety measures, modifying rules to accommodate needs of the growing population etc.

Data includes details on application/permit numbers, job addresses, supervisorial districts, and the current status of the applications.

For the purpose of data imputation techniques, we'll consider 3 rows with 2 numerical and 1 categorical data type.

The Numerical data :- Proposed Unit, Estimated cost

The ordinal data :- Proposed Usage

```
   print(permits['Proposed Units'])

   total_cells = np.product(permits['Proposed Units'].shape)
   total_missing = missing_values_count['Proposed Units'].sum()

   print('\nNUll data percentage',(total_missing/total_cells) * 100)
0          NaN
1          NaN
2         39.0
3          1.0
4          NaN
          ...
198895     NaN
198896     4.0
198897     NaN
198898     NaN
198899     NaN
Name: Proposed Units, Length: 198900, dtype: float64

NUll data percentage 25.596279537456006
```

It can be seen that 25.5 % of the data are missing from **proposed unit** data and 12 % data are missing from **proposed Usage** data.

```
   print(permits['Proposed Use'])

   total_cells = np.product(permits['Proposed Use'].shape)
   total_missing = missing_values_count['Proposed Use'].sum()

   print('\nNUll data percentage',(total_missing/total_cells) * 100)
0                       NaN
1                       NaN
2              retail sales
3         1 family dwelling
4                       NaN
                 ...
198895                  NaN
198896           apartments
198897                  NaN
198898                  NaN
198899                  NaN
Name: Proposed Use, Length: 198900, dtype: object

NUll data percentage 21.33685268979387
```

Data Imputation techniques :-

Complete case analysis :-

```
print(permits['Street Number Suffix'])
total_cells = np.product(permits['Street Number Suffix'].shape)
total_missing = missing_values_count['Street Number Suffix'].sum()
(total_missing/total_cells) * 100
```

```
0         NaN
1         NaN
2         NaN
3         NaN
4         NaN
         ...
198895    NaN
198896    NaN
198897    NaN
198898    NaN
198899    NaN
Name: Street Number Suffix, Length: 198900, dtype: object

98.88587229763701
```

The above diagram shows that, in the Street number suffix attributes, about 98% of the data is missing, if we try complete case analysis to this problem, then almost all of the data would be lost from the data set.

```
permits['Street Number Suffix'].dropna()
```

```
92        A
146       C
164       A
200       C
273       A
         ..
198762    B
198771    A
198772    B
198803    A
198851    A
Name: Street Number Suffix, Length: 2216, dtype: object
```

Here, it can be seen that the number of data falls from 198900 to 2216. Which almost erases every data.

On the other hand, if we only consider the columns that has all of it's records, then we are only left with 13 columns, which drastically fell from 43.

```python
columns_with_na_dropped = permits.dropna(axis=1)
columns_with_na_dropped.head()
```

	Permit Number	Permit Type	Permit Type Definition	Permit Creation Date	Block	Lot	Street Number	Street Name	Current Status	Current Status Date	Filed Date	Record ID
0	201505065519	4	sign - erect	05/06/2015	0326	023	140	Ellis	expired	12/21/2017	05/06/2015	1380611233945
1	201604195146	4	sign - erect	04/19/2016	0306	007	440	Geary	issued	08/03/2017	04/19/2016	1420164406718
2	201605278609	3	additions alterations or repairs	05/27/2016	0595	203	1647	Pacific	withdrawn	09/26/2017	05/27/2016	1424856504716
3	201611072166	8	otc alterations permit	11/07/2016	0156	011	1230	Pacific	complete	07/24/2017	11/07/2016	1443574295566
4	201611283529	6	demolitions	11/28/2016	0342	001	950	Market	issued	12/01/2017	11/28/2016	144548169992

Hence we use other imputation methods to fill the place of missing values in the data.

Numerical Data Imputation :-

Ibuilt imputer from sklearn:-

This imputer from sklearn using deep learning methods to analyse the data and imputes the missing value.

For our case, we're using Proposed units and estimated cost to impute the data.

```python
subset_nfl_data = permits.loc[:, ['Proposed Units','Estimated Cost',] ]
subset_nfl_data
```

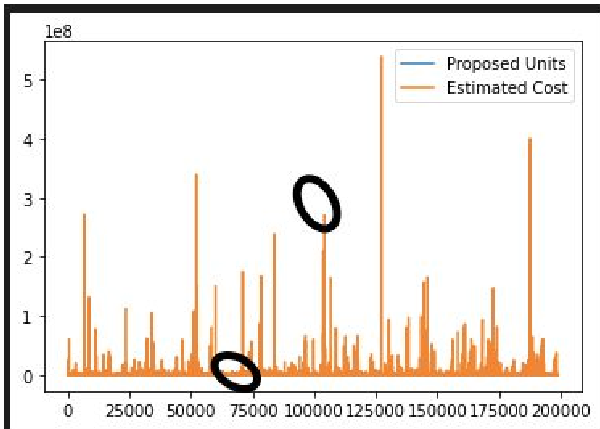
	Proposed Units	Estimated Cost
0	NaN	4000.0
1	NaN	1.0
2	39.0	20000.0
3	1.0	2000.0
4	NaN	100000.0
...	...	...
198895	NaN	NaN
198896	4.0	5000.0
198897	NaN	NaN
198898	NaN	NaN
198899	NaN	NaN

198900 rows × 2 columns

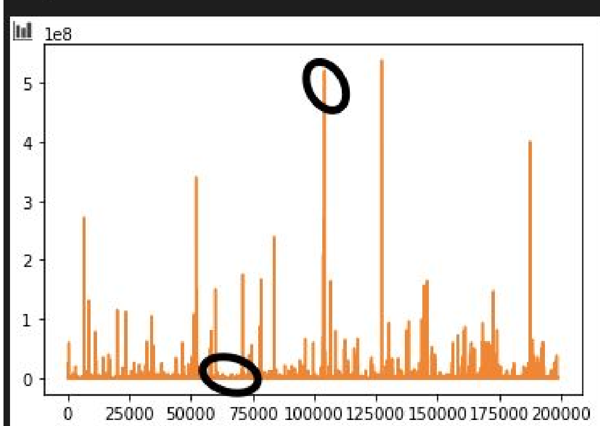
```
my_imputer = SimpleImputer()
data_with_imputed_values = my_imputer.fit_transform(subset_nfl_data)
print(data_with_imputed_values)
```

```
[[1.65109501e+01 4.00000000e+03]
 [1.65109501e+01 1.00000000e+00]
 [3.90000000e+01 2.00000000e+04]
 ...
 [1.65109501e+01 1.68955443e+05]
 [1.65109501e+01 1.68955443e+05]
 [1.65109501e+01 1.68955443e+05]]
```

It can be seen from above two diagrams that, data is imputed by simple imputer and the Null values are filled with decent values that do not affect the model.




```
import matplotlib.pyplot as plt
plt.plot(data_with_imputed_values)
plt.show()
```



Here it can be seen that data values are imputed and missing values are replaced with reasonable values.

Here at around 10L in x axis, it can be seen that a new value is added, which is kind of similar to the max value in the estimated cost. These value may have been unregistered because it may attract large tax rates.

Near the 75K mark, there are few values that are imputed and it's all replaced with small values too because the area where the work goes may bea a village and hence estimates is low.

Mean Data imputation :-

```
permits['Proposed Units']

0          NaN
1          NaN
2         39.0
3          1.0
4          NaN
          ...
198895     NaN
198896     4.0
198897     NaN
198898     NaN
198899     NaN
Name: Proposed Units, Length: 198900, dtype: float64
```

```
me = permits['Proposed Units'].fillna(permits['Proposed Units'].mean())
print (me)

0         16.51095
1         16.51095
2         39.00000
3          1.00000
4         16.51095
            ...
198895    16.51095
198896     4.00000
198897    16.51095
198898    16.51095
198899    16.51095
Name: Proposed Units, Length: 198900, dtype: float64
```

```
print(scores)
184784.40251186382
```

For the proposed Unit section, We use mean values to fill the missing data and it can be seen that we get score of 184784 from using sklearn regression model. Although this model works, we cannot assume that the model is consistent, as even though the data in number 4 has only 1 as the value, the data in 2 is ,made as 16.5 which is the mean value. This could completely mis lead us to assumption that the proposed number of units is always around 16.

In real, the number of units is city is around 40 while, in village it is around 10. But considering it as a overall image, it works fine.

Ordinal Data Imputation :-

Ordinal data can be converted to numerical data and then we can use the same imputations used for nominal data. But here we use frequent data imputation method.

Frequent Category Imputation :-

```
print(permits['Proposed Use'])

0                         NaN
1                         NaN
2                retail sales
3          1 family dwelling
4                         NaN
                  ...
198895                    NaN
198896             apartments
198897                    NaN
198898                    NaN
198899                    NaN
Name: Proposed Use, Length: 198900, dtype: object
```

```
permits['Proposed Use'].fillna(permits['Proposed Use'].value_counts().index[0])

0          1 family dwelling
1          1 family dwelling
2               retail sales
3          1 family dwelling
4          1 family dwelling
                  ...
198895     1 family dwelling
198896            apartments
198897     1 family dwelling
198898     1 family dwelling
198899     1 family dwelling
Name: Proposed Use, Length: 198900, dtype: object
```

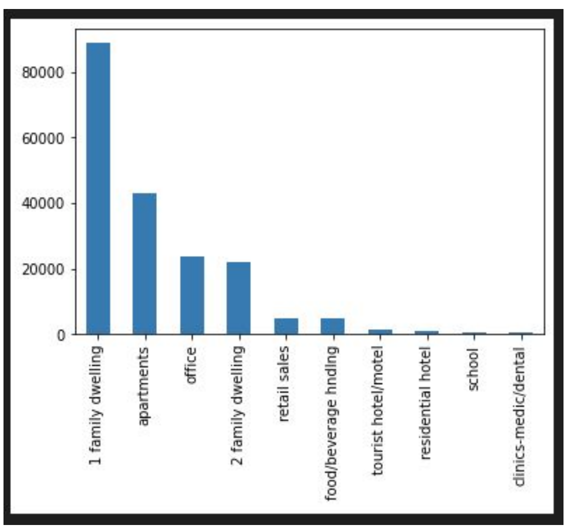
In frequent category imputation, we replace the NaN value with the most frequent values in the dataset.

Here, we can see that Nan values are replaced with 1 family dwelling. It says that most of the permits to build home has been allotted to provide home to the homeless and specifically to single small families.

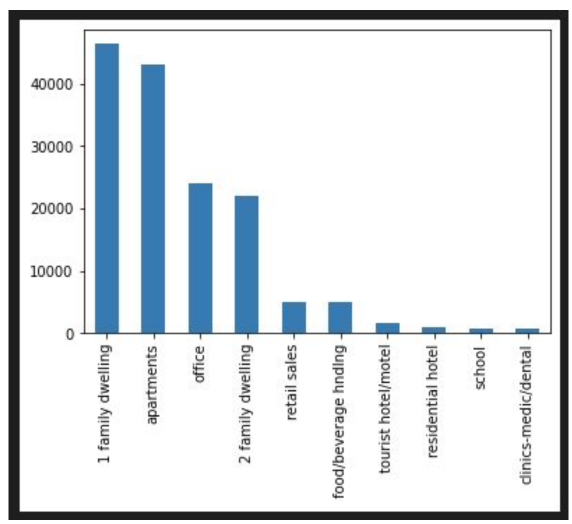
```
Before frequent data imputation

1 family dwelling        46346
apartments               43032
office                   23962
2 family dwelling        22061
retail sales              5079
food/beverage hndlng      5053
tourist hotel/motel       1601
Name: Proposed Use, dtype: int64

After frequent data imputation

1 family dwelling        88785
apartments               43032
office                   23962
2 family dwelling        22061
retail sales              5079
food/beverage hndlng      5053
tourist hotel/motel       1601
Name: Proposed Use, dtype: int64
```

This table shows that 1 family dwelling has considerably raised from 46k to 88k.

Since it is the most frequently repeated element here, the Values are replaced with it.



A



B

The image A shows the imputed data, while B shows the non imputed data, But here imputation makes the model biased to one side. Although it makes the model ready to use. It may not be a good imputation.

Missing Category Imputation :-

```
0         MISSING
1         MISSING
2            39.0
3             1.0
4         MISSING
           ...
198895    MISSING
198896        4.0
198897    MISSING
198898    MISSING
198899    MISSING
Name: Proposed Units, Length: 198900, dtype: object
```

In case of missing value imputation, The “MISSING” is itself added to the missing data.

Here, it can be seen that MISSING values are added to the data. This would not disturb the model and leave the model as it is. Since the amount of data is enormous in our case, and the missing data is not highly significant, Missing category imputation would do nicely in our case.

Thus we have discussed 4 data imputation techniques for the Building permits data and also arrived at best ways to that.