JN

Project - house prices advanced regression techniques

```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

I. Overall understanding of the data

1. Load the data as a pandas DataFrame.
2. Display:
    - The first 5 rows of the dataset
    - Number of instances
    - Number of features
    - Feature names
    - Data type of each feature
    - Number of missing values for each feature
3. Check if the data types are correctly identified. (A common situation is that a numeric feature is identified as "object")
4. Handle missing values. There is no standard procedure of missing value imputation. For simplicity, follow the procedure below:
    - Remove the feature if more than 30% of its values are missing
    - Remove the rows containing the missing values if less than 5% of values are missing in a column
    - If the percentage of missing values is between 5% and 30%, fill the missing data with the most frequent value (categorical feature) or the average value (for numeric feature).

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
#1. Down loaded the Ames Housing Dataset from (https://www.kaggle.com/c/house-prices-a
train_df = pd.read_csv('Data/house-prices-advanced-regression-techniques/train.csv', s
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# 2a.The first 5 rows of the dataset
train_df.head(5)
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Lan
Id								
1	60	RL	65.0	8450	Pave	NaN	Reg	
2	20	RL	80.0	9600	Pave	NaN	Reg	
3	60	RL	68.0	11250	Pave	NaN	IR1	
4	70	RL	60.0	9550	Pave	NaN	IR1	
5	60	RL	84.0	14260	Pave	NaN	IR1	

5 rows × 80 columns

```
# 2b.Number of instances that show the number of 1460 rows and 80 columns
train_df.shape
```

```
(1460, 80)
```

```
# 2c.Number of features
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1460 entries, 1 to 1460
Data columns (total 80 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MSSubClass     1460 non-null   int64  
 1   MSZoning       1460 non-null   object 
 2   LotFrontage    1201 non-null   float64
 3   LotArea        1460 non-null   int64  
 4   Street         1460 non-null   object 
 5   Alley          91 non-null     object 
 6   LotShape       1460 non-null   object 
 7   LandContour    1460 non-null   object 
 8   Utilities      1460 non-null   object 
 9   LotConfig      1460 non-null   object 
 10  LandSlope      1460 non-null   object 
 11  Neighborhood   1460 non-null   object 
 12  Condition1     1460 non-null   object 
 13  Condition2     1460 non-null   object 
 14  BldgType       1460 non-null   object 
 15  HouseStyle     1460 non-null   object 
 16  OverallQual    1460 non-null   int64  
 17  OverallCond    1460 non-null   int64  
 18  YearBuilt      1460 non-null   int64  
 19  YearRemodAdd   1460 non-null   int64  
 20  RoofStyle      1460 non-null   object 
```

```
 21  RoofMatl       1460 non-null   object
 22  Exterior1st    1460 non-null   object
 23  Exterior2nd    1460 non-null   object
 24  MasVnrType     1452 non-null   object
 25  MasVnrArea     1452 non-null   float64
 26  ExterQual      1460 non-null   object
 27  ExterCond      1460 non-null   object
 28  Foundation     1460 non-null   object
 29  BsmtQual       1423 non-null   object
 30  BsmtCond       1423 non-null   object
 31  BsmtExposure   1422 non-null   object
 32  BsmtFinType1   1423 non-null   object
 33  BsmtFinSF1     1460 non-null   int64
 34  BsmtFinType2   1422 non-null   object
 35  BsmtFinSF2     1460 non-null   int64
 36  BsmtUnfSF      1460 non-null   int64
 37  TotalBsmtSF    1460 non-null   int64
 38  Heating        1460 non-null   object
 39  HeatingQC      1460 non-null   object
 40  CentralAir     1460 non-null   object
 41  Electrical     1459 non-null   object
 42  1stFlrSF       1460 non-null   int64
 43  2ndFlrSF       1460 non-null   int64
 44  LowQualFinSF   1460 non-null   int64
 45  GrLivArea      1460 non-null   int64
 46  BsmtFullBath   1460 non-null   int64
 47  BsmtHalfBath   1460 non-null   int64
 48  FullBath       1460 non-null   int64
 49  HalfBath       1460 non-null   int64
 50  BedroomAbvGr   1460 non-null   int64
 51  KitchenAbvGr   1460 non-null   int64
 52  KitchenQual    1460 non-null   object
 53  TotRmsAbvGrd   1460 non-null   int64
```

```
# 2d.Feature names
train_df.columns.values
```

```
array(['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
       'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond',
       'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl',
       'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea',
       'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
       'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC',
       'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
       'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu',
       'GarageType', 'GarageYrBlt', 'GarageFinish', 'GarageCars',
       'GarageArea', 'GarageQual', 'GarageCond', 'PavedDrive',
       'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'PoolArea', 'PoolQC', 'Fence', 'MiscFeature',
       'MiscVal', 'MoSold', 'YrSold', 'SaleType', 'SaleCondition',
       'SalePrice'], dtype=object)
```

```
# 2e.Data type of each feature
print(train_df.dtypes)
```

```
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
Street            object
                  ...
MoSold             int64
YrSold             int64
SaleType          object
SaleCondition     object
SalePrice          int64
Length: 80, dtype: object
```

```
# 2f.Number of missing values for each feature
train_df.isna().sum()
```

```
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
Street             0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 80, dtype: int64
```

3. Check if the data types are correctly identified.

(A common situation is that a numeric feature is identified as "object")

```
# 3a. displaying full dataset columns and checking 3 rows to see if their input values
pd.options.display.max_columns = 100
train_df.head(3)
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	Lan
Id								
1	60	RL	65.0	8450	Pave	NaN	Reg	
2	20	RL	80.0	9600	Pave	NaN	Reg	
3	60	RL	68.0	11250	Pave	NaN	IR1	

```
# 3b. the train_df has object(43) and after reviewing the dataset above it is correct
```

```
# 3b. the train_df has object(43) and after reviewing the dataset above it is correct
# creating a filter to check for objects in train_df
filter = train_df.dtypes
print(filter[filter == np.dtype('object')])
```

```
MSZoning         object
Street           object
Alley            object
LotShape         object
LandContour      object
Utilities        object
LotConfig        object
LandSlope        object
Neighborhood     object
Condition1       object
Condition2       object
BldgType         object
HouseStyle       object
RoofStyle        object
RoofMatl         object
Exterior1st      object
Exterior2nd      object
MasVnrType       object
ExterQual        object
ExterCond        object
Foundation       object
BsmtQual         object
BsmtCond         object
BsmtExposure     object
BsmtFinType1     object
BsmtFinType2     object
Heating          object
HeatingQC        object
CentralAir       object
Electrical       object
KitchenQual      object
Functional       object
FireplaceQu      object
GarageType       object
GarageFinish     object
GarageQual       object
GarageCond       object
PavedDrive       object
PoolQC           object
Fence            object
MiscFeature      object
SaleType         object
SaleCondition    object
dtype: object
```

4. Handle missing values. There is no standard procedure of missing value imputation. For simplicity, follow the procedure below:
    - Remove the feature if more than 30% of its values are missing

- Remove the rows containing the missing values if less than 5% of values are missing in a column
- If the percentage of missing values is between 5% and 30%, fill the missing data with the most frequent value (categorical feature) or the average value (for numeric feature).

```
## Getting the percentage of missing values
total = train_df.isnull().sum().sort_values(ascending=False) #adding up isnull values
percent = (train_df.isnull().sum()/len(train_df) * 100).sort_values(ascending=False)#
missing_data = pd.concat([total, percent], axis=1, keys=['total', 'percent'])
missing_data.head(50) #displaying 23 to see where it ends and reaches 00
```

	total	percent
PoolQC	1453	99.520548
MiscFeature	1406	96.301370
Alley	1369	93.767123
Fence	1179	80.753425
FireplaceQu	690	47.260274
LotFrontage	259	17.739726
GarageType	81	5.547945
GarageCond	81	5.547945
GarageFinish	81	5.547945
GarageQual	81	5.547945
GarageYrBlt	81	5.547945
BsmtFinType2	38	2.602740
BsmtExposure	38	2.602740
BsmtQual	37	2.534247
BsmtCond	37	2.534247
BsmtFinType1	37	2.534247
MasVnrArea	8	0.547945
MasVnrType	8	0.547945
Electrical	1	0.068493
RoofMatl	0	0.000000
Exterior1st	0	0.000000
RoofStyle	0	0.000000
ExterQual	0	0.000000
Exterior2nd	0	0.000000
YearBuilt	0	0.000000
ExterCond	0	0.000000

```
# Remove the feature if more than 30% of its values are missing which are ['PoolQC',

train_df = train_df.drop(columns=['PoolQC', 'MiscFeature', 'Alley', 'Fence','Fireplac
train_df.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContou
Id							
1	60	RL	65.0	8450	Pave	Reg	L
2	20	RL	80.0	9600	Pave	Reg	L
3	60	RL	68.0	11250	Pave	IR1	L
4	70	RL	60.0	9550	Pave	IR1	L
5	60	RL	84.0	14260	Pave	IR1	L

```
# Remove the rows containing the missing values if less than 5% of values are missing
# 5% of 80 columns = 4

#checking for nan values in rows
is_NaN = train_df.isnull()
row_has_NaN = is_NaN.any(axis=1)
rows_with_NaN = train_df[row_has_NaN]
rows_with_NaN.head() #double checking to make sure it worked. 'Yes it did.'
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContou
Id							
8	60	RL	NaN	10382	Pave	IR1	L
13	20	RL	NaN	12968	Pave	IR2	L
15	20	RL	NaN	10920	Pave	IR1	L
17	20	RL	NaN	11241	Pave	IR1	L
18	90	RL	72.0	10791	Pave	Reg	L

```
#creating a colum  to add how many nan columns they are for each row.
nan_row = train_df.isnull().sum(axis=1)
train_df['nan_row'] = train_df.isnull().sum(axis=1)
#dropping row if more <4 nan values appear.
train_df.drop(train_df[train_df.nan_row < 4].index, inplace = True)
train_df.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContou
Id							
18	90	RL	72.0	10791	Pave	Reg	L
89	50	C (all)	105.0	8470	Pave	IR1	L

```
#checking new len value from removals.
len(train_df)
```

```
111
```

```
train_df.head(5)
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContou
Id							
18	90	RL	72.0	10791	Pave	Reg	L
40	90	RL	65.0	6040	Pave	Reg	L
49	190	RM	33.0	4456	Pave	Reg	L
79	90	RL	72.0	10778	Pave	Reg	L
89	50	C (all)	105.0	8470	Pave	IR1	L

```
# If the percentage of missing values is between 5% and 30%, fill the missing data wit
# NOTES: fill in all the columns with the average for every null value & used most use

# creating a function to look at categorical_values to count and to see which are most
def categorical_values(column):
    return train_df[column].value_counts()

print(categorical_values('GarageQual'),categorical_values('GarageFinish'),categorical_
      ,categorical_values('BsmtQual'),categorical_values('BsmtCond'),categorical_value
```

```
TA    30
Name: GarageQual, dtype: int64 Unf    25
RFn     3
Fin     2
Name: GarageFinish, dtype: int64 TA    28
Fa     2
Name: GarageCond, dtype: int64 Detchd     18
Attchd      9
CarPort     2
BuiltIn     1
Name: GarageType, dtype: int64 TA    54
Gd    17
Fa     3
Name: BsmtQual, dtype: int64 TA    64
```

```
Fa      7
Gd      2
Po      1
Name: BsmtCond, dtype: int64 No    57
Av      8
Gd      6
Mn      3
Name: BsmtExposure, dtype: int64 Unf    34
ALQ    11
GLQ    10
Rec     7
BLQ     7
LwQ     5
Name: BsmtFinType1, dtype: int64 Unf    70
GLQ     2
BLQ     1
Rec     1
Name: BsmtFinType2, dtype: int64
```

```
#getting the mean for numeric feature ['GarageYrBlt' &'LotFrontage']
train_df[['GarageYrBlt','LotFrontage']].mean()
```

```
GarageYrBlt    1965.900000
LotFrontage      61.755102
dtype: float64
```

```
# Imputing the missing values that can be either categorical or numeric
def cat_imputation(column, value):
    train_df.loc[train_df[column].isnull(),column] = value

# empty fields here are replaced with most used
cat_imputation('GarageQual','Unf')
cat_imputation('GarageFinish','TA')
cat_imputation('GarageCond','Detchd')
cat_imputation('GarageType','TA')

cat_imputation('BsmtQual','TA')
cat_imputation('BsmtCond','TA')
cat_imputation('BsmtExposure','No')
cat_imputation('BsmtFinType1','Unf')
cat_imputation('BsmtFinType2','Unf')




# empty fields here are replaced with mean value
cat_imputation('GarageYrBlt',1965.900000)
cat_imputation('LotFrontage',61.755102)


#checking if replacement worked succesfully. 'yes it did'
checking = ['GarageQual', 'GarageFinish','GarageCond','GarageType','GarageYrBlt','Lot]
checking1 = train_df[checking]
checking1.head(5)
```

	GarageQual	GarageFinish	GarageCond	GarageType	GarageYrBlt	LotFronta
Id						
18	TA	Unf	TA	CarPort	1967.0	7
40	TA	RFn	Fa	Detchd	1965.9	6
49	TA	RFn	Fa	Detchd	1965.9	3
79	TA	RFn	Fa	Detchd	1965.9	7
89	TA	RFn	Fa	Detchd	1965.9	10

```
print(train_df.isna().sum().values)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0]
```

Section II Most relevant features

[ ] ↳ *12 cells hidden*

III: Bivariate analysis

For each of the four chosen predictive features:

1. Draw the scatter plot of this feature against SalePrice (set title, axis label properly).
2. Calculate the correlation coefficient
3. Describe what you discover: positive correlation, negative correlation, no correlation

[ ] ↳ *6 cells hidden*

CMP 464 Midterm Project: Predicting Housing Prices at Ames, Iowa, Part II

IV. Identify additional relevant feature Previously we have found 4 features that are useful for predicting the sale price. Let’s find out if there are other features that we should consider as well.

1. Find features with high correlation: For each numeric feature, calculate its correlation coefficient with feature SalePrice. Identify the feature (other than aforementioned 5 features) that has the strongest correlation with the sale prices.

2. Feature engineering: Based on our experience, the total area of the house and the average area per room should also be important factors in determining the price. Please create these two columns using the following formula:

```
1) total area = total area above ground (“GrLivArea”) + total basement area (“TotalBsmtSF”)
2) area per room = total area above ground (“GrLivArea”) / number of rooms (“TotRmsAbvGrd”).
```

At this point, we have selected 7 features that are helpful to predict the sale price: “OverallQual”, “YearBuilt”, “TotalBasmtSF”, “GrLivArea”, Feature selected in IV.1, “TotalArea”, “AreaPerRoom”.

[ ] ↳ *6 cells hidden*

V. Prepare data for k-Nearest-Neighbor method.

1. Create a new data frame with SalePrice and the 7 selected features.
2. For each of the 7 selected features, calculate its standard deviation. These values will be used as weights for the k-nearest-neighbor prediction.

[ ] ↳ *6 cells hidden*

VI. Apply the kNN (k=5) method to predict sale price of the first instance from the test set.

For first instance in test.csv, predict its price using the k-nearest-neighbor method. Consider using the following procedure:

1. Load test.csv and extract its first row. Calculate its total area and area per room.

2. Add a new column (“Diff”) to the data frame representing the training data. For each row, calculate a weighted sum of the differences between this row’s features and those from the test row. For each feature, use the reciprocal of its standard deviation as its weight.

3. Sort the rows so that values in the “Diff” column is listed in ascending order. The top 5 instances are the closest neighbors of the new instance y.

4. Calculate the average sale price of these 5 instances. This will be the prediction of the new instance.

[ ] ↳ *6 cells hidden*