Applied Data Science with Python

Certification Project

Author: Charusmita Shah

Feature Engineering

Objectives:

- The aim of the project is to help understand working with the dataset and performing analysis.
- This project will assess the data and prepares a fresh dataset for training and prediction
- To create a box plot to identify the variables with outliers

Project Statement:

While searching for the dream house, the buyer looks at various factors, not just at the height of the basement ceiling or the proximity to an east-west railroad.
Using the dataset, find the factors that influence price negotiations while buying a house.
There are 79 explanatory variables describing every aspect of residential homes in Ames, Iowa.

Dataset Description:

Variable	Description
SalePrice	The property's sale price is in dollars. This is the target variable that you're trying to predict.
MSSubClass	The building class
MSZoning	The general zoning classification
LotFrontage	Linear feet of street connected to property
LotArea	Lot size in square feet
Street	Type of road access
Alley	Type of alley access
LotShape	General shape of property
LandContour	Flatness of the property
Utilities	Type of utilities available
LotConfig	Lot configuration
LandSlope	Slope of property
Neighborhood	Physical locations within Ames city limits

Condition1	Proximity to main road or railroad
Condition2	Proximity to main road or railroad (if a second is present)
BldgType	Type of dwelling
HouseStyle	Style of dwelling
OverallQual	Overall material and finish quality
OverallCond	Overall condition rating
YearBuilt	Original construction date
YearRemodAdd	Remodel date
RoofStyle	Type of roof
RoofMatl	Roof material
Exterior1st	Exterior covering on house
Exterior2nd	Exterior covering on house (if more than one material)
MasVnrType	Masonry veneer type
MasVnrArea	Masonry veneer area in square feet
ExterQual	Exterior material quality
ExterCond	Present condition of the material on the exterior
Foundation	Type of foundation
BsmtQual	Height of the basement
BsmtCond	General condition of the basement
BsmtExposure	Walkout or garden level basement walls
BsmtFinType1	Quality of the basement finished area
BsmtFinSF1	Type 1 finished square feet
BsmtFinType2	Quality of second finished area (if present)
BsmtFinSF2	Type 2 finished square feet
BsmtUnfSF	Unfinished square feet of basement area
TotalBsmtSF	Total square feet of basement area
Heating	Type of heating
HeatingQC	Heating quality and condition
CentralAir	Central air conditioning
Electrical	Electrical system
1stFlrSF	First Floor square feet
2ndFlrSF	Second floor square feet
LowQualFinSF	Low quality finished square feet (all floors)
GrLivArea	Above grade (ground) living area square feet

BsmtFullBath	Basement full bathrooms
BsmtHalfBath	Basement half bathrooms
FullBath	Full bathrooms above grade
HalfBath	Half bathrooms above grade
Bedroom	Number of bedrooms above basement level
Kitchen	Number of kitchens
KitchenQual	Kitchen quality
TotRmsAbvGrd	Total rooms above grade (does not include bathrooms)
Functional	Home functionality rating
Fireplaces	Number of fireplaces
FireplaceQu	Fireplace quality
GarageType	Garage location
GarageYrBlt	Year garage was built
GarageFinish	Interior finish of the garage
GarageCars	Size of the garage in car capacity
GarageArea	Size of the garage in square feet
GarageQual	Garage quality
GarageCond	Garage condition
PavedDrive	Paved driveway
WoodDeckSF	Wood deck area in square feet
OpenPorchSF	Open porch area in square feet
EnclosedPorch	Enclosed porch area in square feet
3SsnPorch	Three season porch area in square feet
ScreenPorch	Screen porch area in square feet
PoolArea	Pool area in square feet
PoolQC	Pool quality
Fence	Fence quality
MiscFeature	Miscellaneous feature not covered in other categories
MiscVal	$Value of miscellaneous feature
MoSold	Month Sold
YrSold	Year Sold
SaleType	Type of sale
SaleCondition	Condition of sale

Note:

1) Download the “PEP1.csv” using the link given in the Feature Engineering project problem statement
2) For a detailed description of the dataset, you can download and refer to data_description.txt using the link given in the Feature Engineering project problem statement

Tasks to Perform (detailed *with Solutions*):

1. Import the necessary libraries

1.1 Pandas is a Python library for data manipulation and analysis.

1.2 NumPy is a package that contains a multidimensional array object and several derivative ones.

1.3 Matplotlib is aPython visualization package for 2D array plots.

1.4 Seaborn is built on top of Matplotlib. It's used for exploratory data analysis and data visualization.

Ans.

```
 import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Read the dataset:

2.1 Understand the dataset

Ans . We read the csv file with the read_csv function from pandas from the specific folder.

```
df = pd.read_csv("PEP1.csv")
df.head()
```

[8]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl

2.2 Print the name of the columns

Ans. `df.columns`

```
[10]: df.columns
```

```
[10]: Index(['Id', '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', 'Bedroom', 'Kitchen', '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')
```

2.3 Print the shape of the dataframe

Ans. `df.shape`

```
[12]: df.shape
```

```
[12]: (1460, 81)
```

2.4 Check for null values

Ans. `df.isnull().sum()`

```
df.isnull().sum()
```

```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     259
LotArea           0
                 ...
MoSold            0
YrSold            0
SaleType          0
SaleCondition     0
SalePrice         0
Length: 81, dtype: int64
```

2.5 Print the unique values

Ans.

```
print(df.apply(lambda col: col.unique()))
```

```
# 2.5 Print the unique values
```

```
print(df.apply(lambda col: col.unique()))
```

```
Id               [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...
MSSubClass       [60, 20, 70, 50, 190, 45, 90, 120, 30, 85, 80,...
MSZoning                                 [RL, RM, C (all), FV, RH]
LotFrontage      [65.0, 80.0, 68.0, 60.0, 84.0, 85.0, 75.0, nan...
LotArea          [8450, 9600, 11250, 9550, 14260, 14115, 10084,...
                                      ...
MoSold                       [2, 5, 9, 12, 10, 8, 11, 4, 1, 7, 3, 6]
YrSold                               [2008, 2007, 2006, 2009, 2010]
SaleType         [WD, New, COD, ConLD, ConLI, CWD, ConLw, Con, ...
SaleCondition    [Normal, Abnorml, Partial, AdjLand, Alloca, Fa...
SalePrice        [208500, 181500, 223500, 140000, 250000, 14300...
Length: 81, dtype: object
```

2.6 Select the numerical and categorical variables

Ans.

```
numeric_data = dataset.select_dtypes(include=[np.number])
categorical_data = dataset.select_dtypes(exclude=[np.number])
```

```
[19]: numeric_data = df.select_dtypes(include=[np.number])
      categorical_data = df.select_dtypes(exclude=[np.number])

[21]: numeric_data.columns

[21]: Index(['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
             'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1',
             'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
             'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
             'HalfBath', 'Bedroom', 'Kitchen', 'TotRmsAbvGrd', 'Fireplaces',
             'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
             'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal',
             'MoSold', 'YrSold', 'SalePrice'],
            dtype='object')

[22]: categorical_data.columns

[22]: Index(['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities',
             'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1', 'Condition2',
             'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st',
```

3) Descriptive stats and EDA

3. EDA of numerical variables

3.2 Missing value treatment

Ans.

By doing

```
numeric_data.isnull().sum()/len(numeric_data)*100,
```

we see the percentage of missing data.

```
]: numeric_data.isnull().sum()/len(numeric_data)*100
```

```
]: Id               0.000000
   MSSubClass       0.000000
   LotFrontage     17.739726
   LotArea          0.000000
   OverallQual      0.000000
   OverallCond      0.000000
   YearBuilt        0.000000
   YearRemodAdd     0.000000
   MasVnrArea       0.547945
   BsmtFinSF1       0.000000
   BsmtFinSF2       0.000000
   BsmtUnfSF        0.000000
   TotalBsmtSF      0.000000
   1stFlrSF         0.000000
   2ndFlrSF         0.000000
   LowQualFinSF     0.000000
   GrLivArea        0.000000
   BsmtFullBath     0.000000
   BsmtHalfBath     0.000000
   FullBath         0.000000
   HalfBath         0.000000
   Bedroom          0.000000
   Kitchen          0.000000
   TotRmsAbvGrd     0.000000
   Fireplaces       0.000000
   GarageYrBlt      5.547945
   GarageCars       0.000000
   GarageArea       0.000000
```

We see: LotFrontage having 17.74% missing data and GarageYrBlt having 5.55% .
1) Missing data lesser than 10% can be safely removed. So dropping na columns from GarageYrBlt.

```
]: numeric_data.GarageYrBlt.unique()
```

```
]: array([2003., 1976., 2001., 1998., 2000., 1993., 2004., 1973., 1931.,
          1939., 1965., 2005., 1962., 2006., 1960., 1991., 1970., 1967.,
          1958., 1930., 2002., 1968., 2007., 2008., 1957., 1920., 1966.,
          1959., 1995., 1954., 1953.,   nan, 1983., 1977., 1997., 1985.,
          1963., 1981., 1964., 1999., 1935., 1990., 1945., 1987., 1989.,
          1915., 1956., 1948., 1974., 2009., 1950., 1961., 1921., 1900.,
          1979., 1951., 1969., 1936., 1975., 1971., 1923., 1984., 1926.,
          1955., 1986., 1988., 1916., 1932., 1972., 1918., 1980., 1924.,
          1996., 1940., 1949., 1994., 1910., 1978., 1982., 1992., 1925.,
          1941., 2010., 1927., 1947., 1937., 1942., 1938., 1952., 1928.,
          1922., 1934., 1906., 1914., 1946., 1908., 1929., 1933.])
```

```
]: # less than 10% missing data can be safely dropped!
   numeric_data = numeric_data[numeric_data['GarageYrBlt'].notna()]
```

2) And for greater than 10%, replacing nan with 0 values. As it means Linear feet of street connected to property(0 means on property and considering as outlier)

```
numeric_data["LotFrontage"] =
numeric_data["LotFrontage"].fillna(0)
```

[41]:
```
# we can assign the nan value to 0 as it means Linear feet of street connected to property
# (0 means on property and considering as outlier)
numeric_data["LotFrontage"] = numeric_data["LotFrontage"].fillna(0)
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.htm
```

[39]:
```
numeric_data.LotFrontage.unique()
```

[39]:
```
array([ 65.,  80.,  68.,  60.,  84.,  85.,  75.,   0.,  51.,  50.,  70.,
        91.,  72.,  66., 101.,  57.,  44., 110.,  98.,  47., 108., 112.,
        74., 115.,  61.,  48.,  33.,  52., 100.,  24.,  89.,  63.,  76.,
        81.,  95.,  69.,  21.,  32.,  78., 121., 122.,  40., 105.,  73.,
        77.,  64.,  94.,  34.,  90.,  55.,  88.,  82.,  71., 120., 107.,
        92., 134.,  62.,  86., 141.,  97.,  54.,  41.,  79., 174.,  99.,
        67.,  83.,  43., 103.,  93.,  30., 129., 140.,  35.,  37., 118.,
        87., 116., 150., 111.,  49.,  96.,  59.,  36.,  56., 102.,  58.,
        38., 109., 130.,  53., 137.,  45., 106., 104.,  42.,  39., 144.,
       114., 128., 149., 313., 168., 182., 138., 160., 152., 124., 153.,
        46.])
```

3.3 Identify the skewness and distribution

Ans. `numeric_data.skew(axis=1, skipna = True)`

[42]:
```
# 3.3 Identify the skewness and distribution
```

[43]:
```
numeric_data.skew(axis = 1, skipna = True)
```

[43]:
```
0       6.146334
1       6.134972
2       6.138704
3       6.116070
4       6.132632
          ...
1455    6.141124
1456    6.125196
1457    6.151234
1458    6.115871
1459    6.117535
Length: 1460, dtype: float64
```

[ ]:

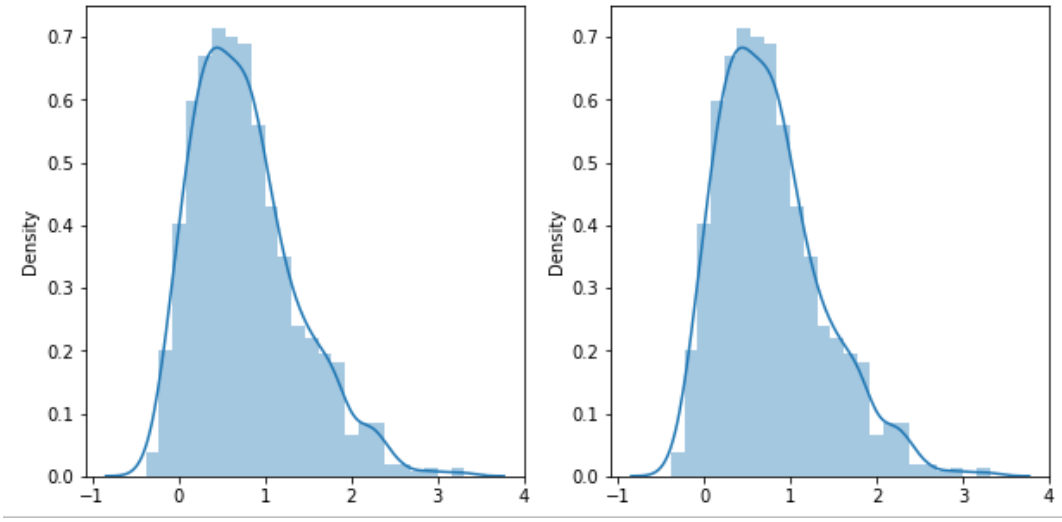
To plot distribution: `fig, ax = plt.subplots(1, 2, figsize=(10, 5))`

```
numeric_data = skewnorm.rvs(size=1000, a=5)
sns.distplot(numeric_data, ax=ax[0])
sns.distplot(numeric_data, ax=ax[1])
```

```
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
numeric_data = skewnorm.rvs(size=1000, a=5)
sns.distplot(numeric_data, ax=ax[0])
sns.distplot(numeric_data, ax=ax[1])
```

```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a
o use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-lev
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a
o use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-lev
  warnings.warn(msg, FutureWarning)
```

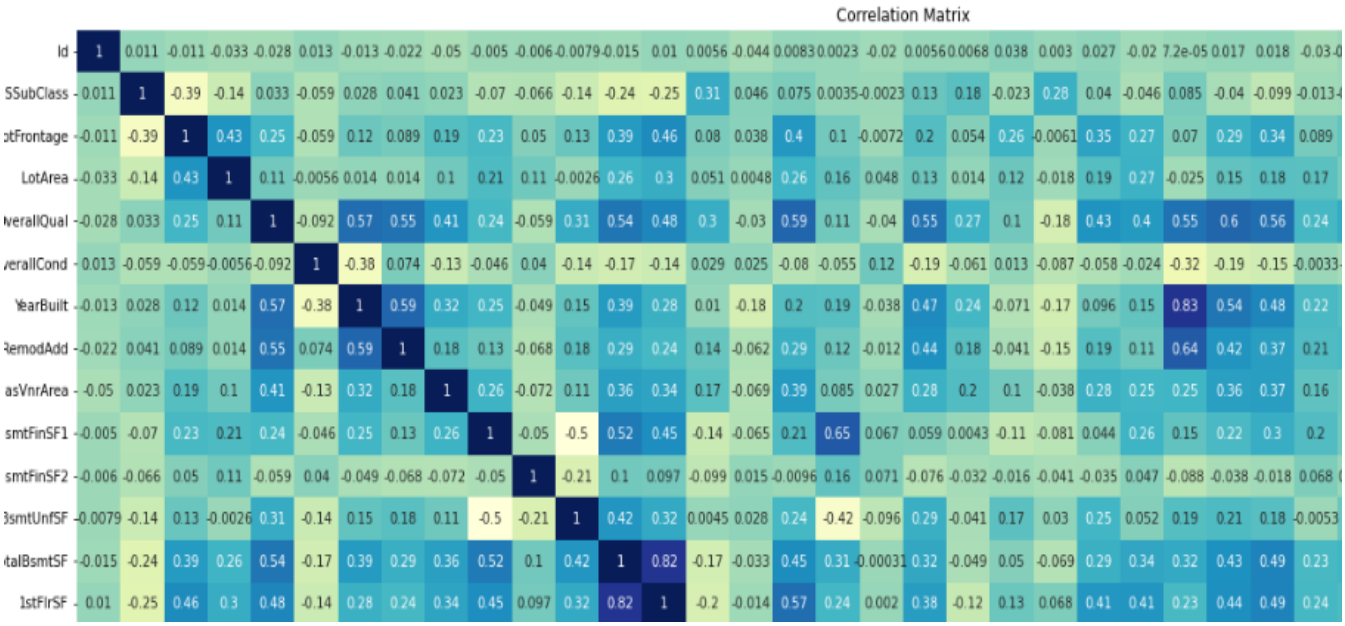
```
<AxesSubplot:ylabel='Density'>
```



3.4 Identify significant variables using a correlation matrix

Ans. The more a column is correlated with other input columns, it can be duplicate. So we need to remove highly correlated columns using correlation matrix.

```
plt.figure(figsize=(30,20))
plt.title("Correlation Matrix")
sns.heatmap(numeric_data.corr(), color="k", annot=True,
cmap="YlGnBu")
```




```
def correlation(dataset,threshold):
    col_corr=set()
    corr_matrix=dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if corr_matrix.iloc[i,j] > threshold:
                colname=corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr

# Correlation coefficients whose magnitude are greater than 0.7
can be considered highly correlated. so dropping them.
col = correlation(numeric_data, 0.7)
```

```
def correlation(dataset,threshold):
    col_corr=set()
    corr_matrix=dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if corr_matrix.iloc[i,j] > threshold:
                colname=corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr
```

```
# Correlation coefficients whose magnitude are greater than 0.7 can be cons
col = correlation(numeric_data, 0.7)
```

```
col
```

```
{'1stFlrSF', 'GarageArea', 'GarageYrBlt', 'SalePrice', 'TotRmsAbvGrd'}
```

Dropping above columns

```
numeric_data.drop(col, axis=1, inplace=True)
```

```
numeric_data.drop(col, axis=1, inplace=True)
```

```
/usr/local/lib/python3.7/site-packages/pandas/core/frame.py:4174: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
  errors=errors,
```

```
numeric_data.columns
```

```
Index(['Id', 'MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual',
       'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1',
       'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '2ndFlrSF', 'LowQualFinSF',
       'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
       'Bedroom', 'Kitchen', 'Fireplaces', 'GarageCars', 'WoodDeckSF',
       'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea',
       'MiscVal', 'MoSold', 'YrSold'],
      dtype='object')
```

3.5 Pair plot for distribution and density

Ans. Pair plot works only on categorical_data. Considering only few columns

```
cols_considered = ["Neighborhood","HouseStyle", "RoofStyle",
"Functional", "GarageType", "SaleType", "SaleCondition"]
# Pair plot for distribution
sns.pairplot(categorical_data, vars=cols_considered,
diag_kind="hist")
```

```
[115]: # Pair plot for distribution
       sns.pairplot(categorical_data, vars=cols_considered, diag_kind="hist")
```

```
[115]: <seaborn.axisgrid.PairGrid at 0x7f1aa46f7550>
```