Data Cleaning and EDA using python

By: Smith Kwabena Agyei

Case Study

Perform EDA and initial Cleaning on the dataset provided. Justify all choices and produce a document to be handed over to the modeling team that includes all plots, charts and deletions.

Steps Taken to Clean Data

After reading the csv file into a dataframe.
Using `df = pd.read_csv('/content/raw_house_data.csv')`

```
I had 5000 rows and 16 columns
```

- ➔ **Checked for duplicates**

  `df.duplicated().sum() “returned 0”`

- ➔ **Viewed the Datatypes of the Dataset**

  df.dtypes

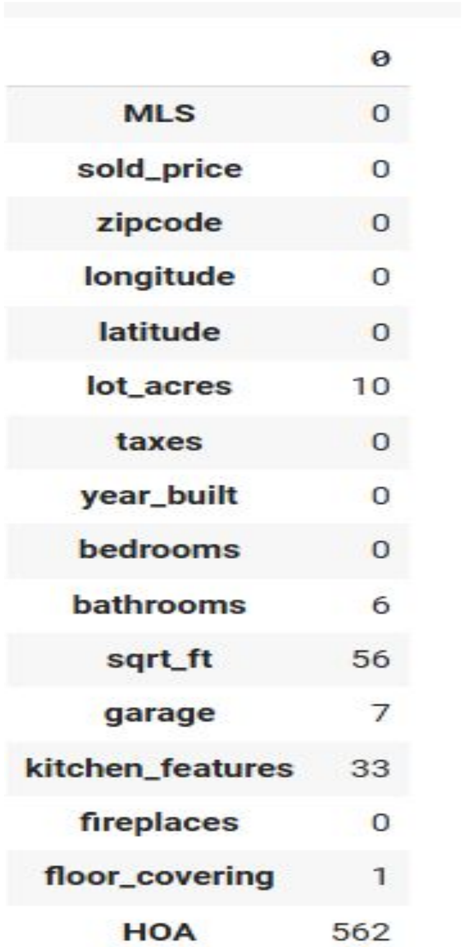
Nulls in Dataset

I checked for the count of null values in all columns.

➔ **df.isnull().sum()**

This code gave me this output which show the count of records/rows with null values based on the columns in the dataset.

Nulls in Dataset cont'd



	0
MLS	0
sold_price	0
zipcode	0
longitude	0
latitude	0
lot_acres	10
taxes	0
year_built	0
bedrooms	0
bathrooms	6
sqrt_ft	56
garage	7
kitchen_features	33
fireplaces	0
floor_covering	1
HOA	562

Remove rows with Lot_acres as null

➔ **Return indexes of Lot_acres columns with null / NaN values**

```
vals = [np.NaN]
mask = df["lot_acres"].isin(vals)
df[mask].index
```

➔ **this removes all records/rows with null value in the Lot_acres column. (This is because the Lot_Acres column is not consistent with the Data) 10 records affected (4990)**

```
df = df.drop(df.index[df[mask].index],
axis=0)
```

Update all columns having null values

➔ Updating some few Columns[Garage,fireplaces,HOA,Kitchen_Features,Floor_Covering] with 0 / "Nothing" where the value is null or empty

```
df["garage"] = df["garage"].fillna(0)
df["HOA"] = df["HOA"].fillna(0)
df["fireplaces"] =
df["fireplaces"].replace(" ", 0)
```

➔ updating Kitchen_features and Floor_covering to Nothing where value is null

```
df["kitchen_features"] =
df["kitchen_features"].replace(np.NaN,
"Nothing")
df["floor_covering"] =
df["floor_covering"].replace(np.NaN,
"Nothing")
df
```

Updating bathroom column with values

➔ **Fixing null / empty values for the bathroom column**

```
zeroBathrooms =
df[df["bathrooms"].isnull()]
#The immediate code snippet assigns
the columns with null values for
bathrooms to variable zeroBathrooms
zeroBathrooms
zBathrmsIndex = zeroBathrooms.index
#The index of the columns is stored in
a variable so that the records can be
accessible using their indexes
zBathrmsIndex
```

Updating bathroom column with values (Cont'd)

➔ **Run a loop to fill the null values of the bathroom columns**

```
for i in zBathrmsIndex:
bathVals = df[(df["bedrooms"] ==
df.loc[i,
"bedrooms"])][["bathrooms"]].mode()
# for every record the mode for
bathrooms that have the same number of
bedrooms is assigned to a variable
bathVals = bathVals["bathrooms"]
# The bathroom column of the bathVals
dataframe is assigned to the variable
df.loc[i, "bathrooms"] = bathVals[0]
# The value of bathVals is assigned to
the respective bathroom columns of the
indexes
print(df.loc[i, "bathrooms"]) # The
update values are printed out.
```

Updating Sqrt_ft column with values

➔ **Fixing null / empty values for the Sqrt_ft column**

```
nSqrft = df[df["sqrt_ft"].isnull()]
#The immediate code snippet assigns
the columns with null values for
bathrooms to variable nSqrft
nSqrft
nSqrftIndex = nSqrft.index
#The index of the columns is stored in
a variable so that the records can be
accessible using their indexes
nSqrftIndex
```

Updating Sqrt_Ft column with values (Cont'd)

➔ **Run a loop to fill the null values of the Sqrt_ft columns**

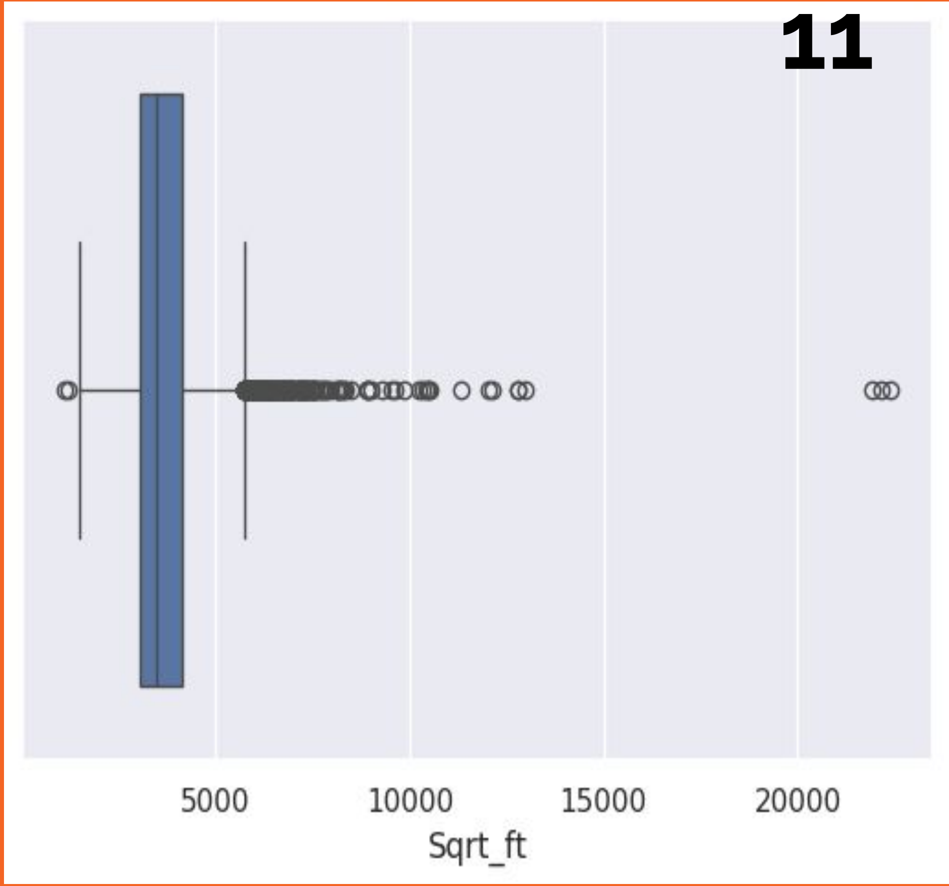
```
for i in nSqrftIndex:
      sqrtftVals = df[(df["bedrooms"]
== df.loc[i,
"bedrooms"])][["sqrt_ft"]].mean() // 1
# for every record the mean for
sqrt_ft that have the same number of
bedrooms is assigned to a variable
sqrtftVals = sqrtftVals["sqrt_ft"]
# The sqrt_ft column of the sqrtftVals
dataframe is assigned to the variable
df.loc[i, "sqrt_ft"] = sqrtftVals #
The value of sqrtftVals is assigned to
the respective sqrt_ft columns of the
indexes
print(sqrtftVals) # The value of
sqrtftVals is assigned to the
respective sqrt_ft columns of the
indexes
```

Aggregation and Renaming Columns

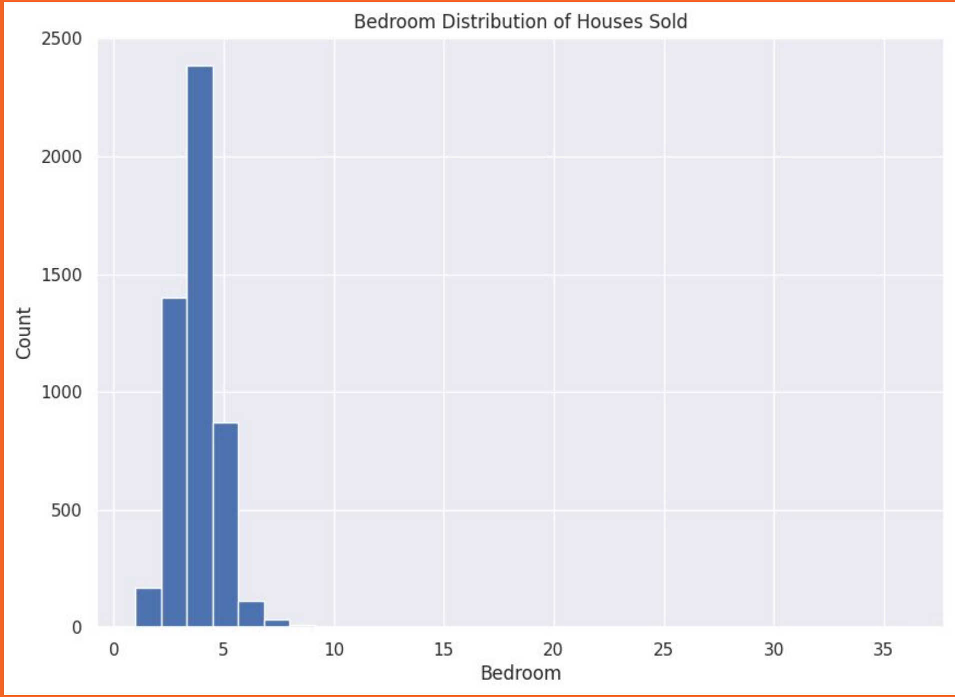
➔ **Run aggregations on the dataframe**

```
df.describe()

#Renaming columns

df = df.rename(columns={'sold_price' :
'Sold_Price', 'zipcode' : 'Zipcode',
'longitutde' : 'Longitutde',
'latitude' : 'Latitude', 'lot_acres' :
'Lot_Acres','taxes' : 'Taxes',
'year_built':'Year_Built', 'bedrooms'
: 'Bedrooms', 'bathrooms' :
'Bathrooms', 'sqrt_ft' : 'Sqrt_ft',
'garage' : 'Garage',
'kitchen_features' :
‘Kitchen_Features', 'fireplaces' :
'Fireplaces', 'floor_covering' :
'Floor_Covering'})
```

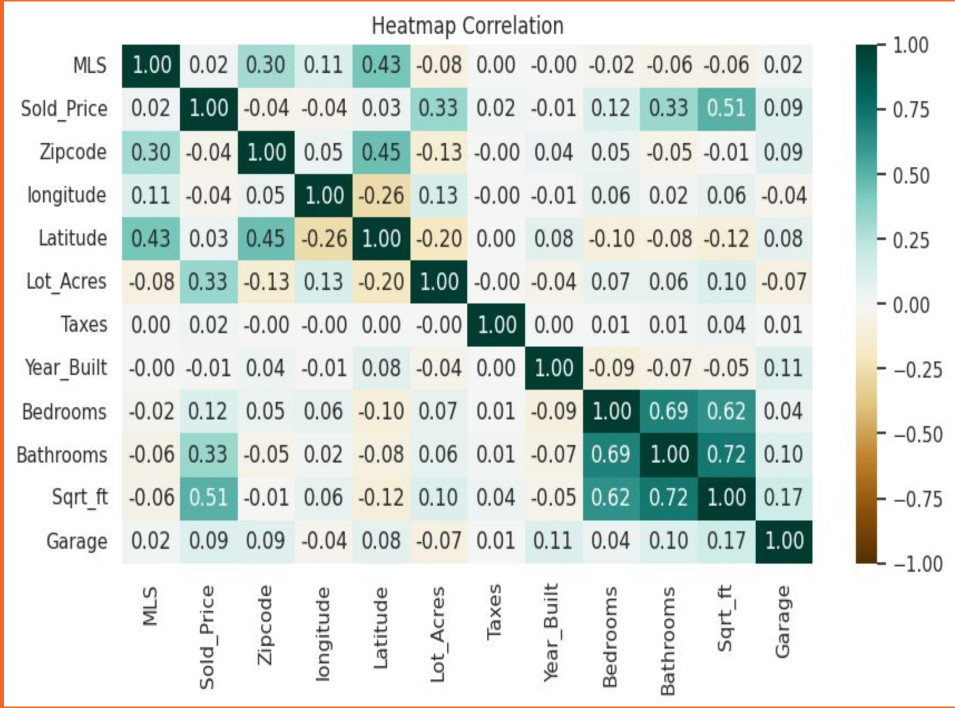
```
sns.boxplot(x=df["Zipcode"])
```



```
plt.figure(figsize=(10,7))
plt.hist(df['Bedrooms'], bins=30)
plt.title('Bedroom Distribution of Houses Sold')
plt.xlabel('Bedroom')
plt.ylabel('Count')
```



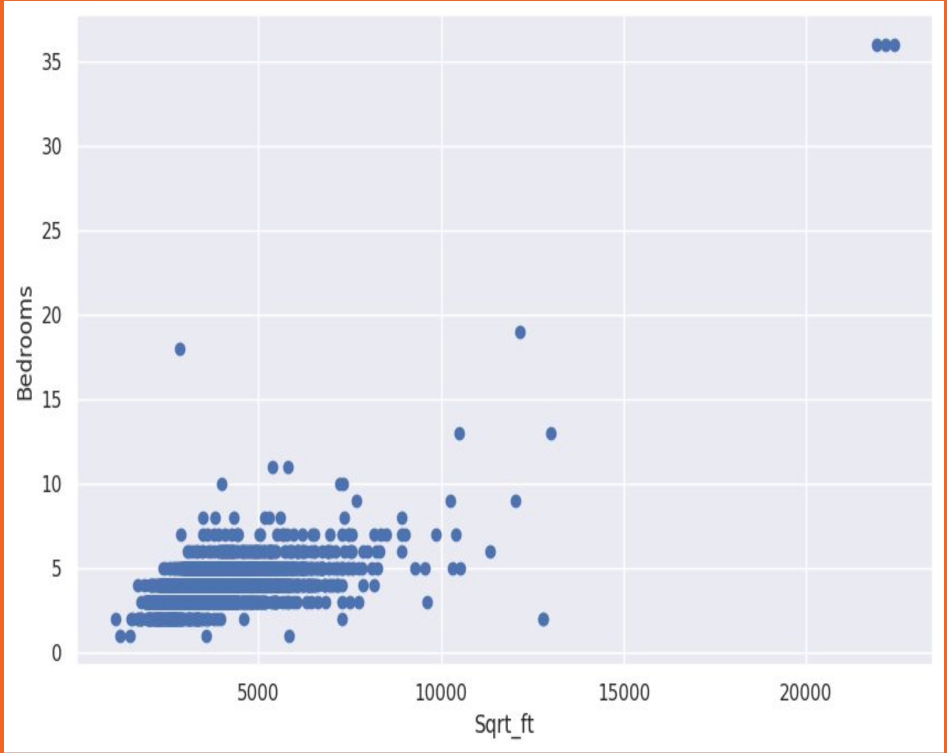
```
plt.figure(figsize=(10,7))
plt.hist(df['Bedrooms'], bins=30)
plt.title('Bedroom Distribution of Houses Sold')
plt.xlabel('Bedroom')
plt.ylabel('Count')
```



	MLS	Sold_Price	Zipcode	longitude	Latitude	Lot_Acres	Taxes	Year_Built	Bedrooms	Bathrooms	Sqrt_ft	Garage
MLS	1.00	0.02	0.30	0.11	0.43	-0.08	0.00	-0.00	-0.02	-0.06	-0.06	0.02
Sold_Price	0.02	1.00	-0.04	-0.04	0.03	0.33	0.02	-0.01	0.12	0.33	0.51	0.09
Zipcode	0.30	-0.04	1.00	0.05	0.45	-0.13	-0.00	0.04	0.05	-0.05	-0.01	0.09
longitude	0.11	-0.04	0.05	1.00	-0.26	0.13	-0.00	-0.01	0.06	0.02	0.06	-0.04
Latitude	0.43	0.03	0.45	-0.26	1.00	-0.20	0.00	0.08	-0.10	-0.08	-0.12	0.08
Lot_Acres	-0.08	0.33	-0.13	0.13	-0.20	1.00	-0.00	-0.04	0.07	0.06	0.10	-0.07
Taxes	0.00	0.02	-0.00	-0.00	0.00	-0.00	1.00	0.00	0.01	0.01	0.04	0.01
Year_Built	-0.00	-0.01	0.04	-0.01	0.08	-0.04	0.00	1.00	-0.09	-0.07	-0.05	0.11
Bedrooms	-0.02	0.12	0.05	0.06	-0.10	0.07	0.01	-0.09	1.00	0.69	0.62	0.04
Bathrooms	-0.06	0.33	-0.05	0.02	-0.08	0.06	0.01	-0.07	0.69	1.00	0.72	0.10
Sqrt_ft	-0.06	0.51	-0.01	0.06	-0.12	0.10	0.04	-0.05	0.62	0.72	1.00	0.17
Garage	0.02	0.09	0.09	-0.04	0.08	-0.07	0.01	0.11	0.04	0.10	0.17	1.00

Created a scatter plot using Matplotlib to visualize the relationship between the two variables `Sqrt_ft` (presumably the square footage of properties) and `Bedrooms` (the number of bedrooms).

```
fig, ax = plt.subplots(figsize=(10, 6))# figure
with dimensions and subplots
ax.scatter(df['Sqrt_ft'], df['Bedrooms'])
#scatter plot is created
plt.title('Bedroom Distribution of Houses Sold')
ax.set_xlabel('Sqrt_ft') label set for x-axis
ax.set_ylabel('Bedrooms') label set for y-axis
```



THANK YOU