Module 1: House Prices: Advanced Regression Techniques EDA

Xinyu Zhao

The question to answer in this project is to identify critical factors that impact the house prices in Ames, Iowa (Dependent variable 'SalePrice'). Aside from this main question, the project may also explore the distributions, correlations, and trends of other independent variables. Finally, the project will utilize various machine learning techniques like regression and random forest, to forecast Saleprice based on independent variables given. "Test" dataset will be used to check model performance. This part of the project will start with an initial exploratory data analysis and then data scaling and comparison.

After loading the training dataset to a dataframe using python pandas package, I can see SalePrice is in integer format. I can also confirm there is no missing data for SalePrice and 1460 records in total.

```
In [2]: df = pd.read_csv("train.csv")

In [6]: df.head()
Out[6]:
```

otArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2008	WD	Normal	208500
9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5	2007	WD	Normal	181500
11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9	2008	WD	Normal	223500
9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2006	WD	Abnorml	140000
14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12	2008	WD	Normal	250000

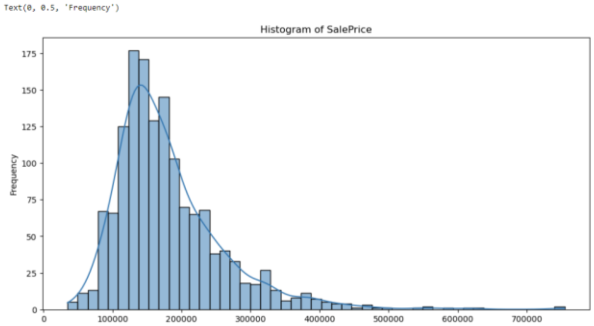
```
<class 'pandas.core.series.Series'>
RangeIndex: 1460 entries, 0 to 1459
Series name: SalePrice
Non-Null Count  Dtype
--------------  -----
1460 non-null   int64
dtypes: int64(1)
memory usage: 11.5 KB
```

```
In [28]: df['SalePrice'].isna().sum()
Out[28]: 0
```

Now I create a histogram and check if there are any outliers. I can see the distribution is skewed to the right which means there are some houses sold at much higher price than median level. The descriptive statistics further strength my belief of outliers. I can see Max value is \$755K significantly higher than 1.5*IQR+third quartile, which is \$340K. Also, Min value is 34900 and is lower than first quartile - 1.5*IQR, which is \$46K. This means there are outliers on both sides of the distribution. For the following analysis, I need to remove/adjust these outliers.

```
plt.figure(figsize=(12, 6))
sns.histplot(df['SalePrice'], kde=True)
plt.title('Histogram of SalePrice')
plt.xlabel('Sale Price')
plt.ylabel('Frequency')
```

```
Text(0, 0.5, 'Frequency')
```




```
print("descriptive stats for SalePrice")
df.describe().applymap(lambda x: f"{x:0.1f}")["SalePrice"]
```

```
descriptive stats for SalePrice

count      1460.0
mean     180921.2
std       79442.5
min       34900.0
25%      129975.0
50%      163000.0
75%      214000.0
max      755000.0
Name: SalePrice, dtype: object
```

Now I am exploring three variables, YearBuilt, Street, and Salecondition. I can see that for YearBuilt, there is a positive relationship of sales price and Yearbuilt. The newer the house, the more expensive. From street variable, I can see house by the Pave Street sell at a higher price than at gravel street. From sale condition, I can see Partial condition sell at significantly higher price than other conditions.

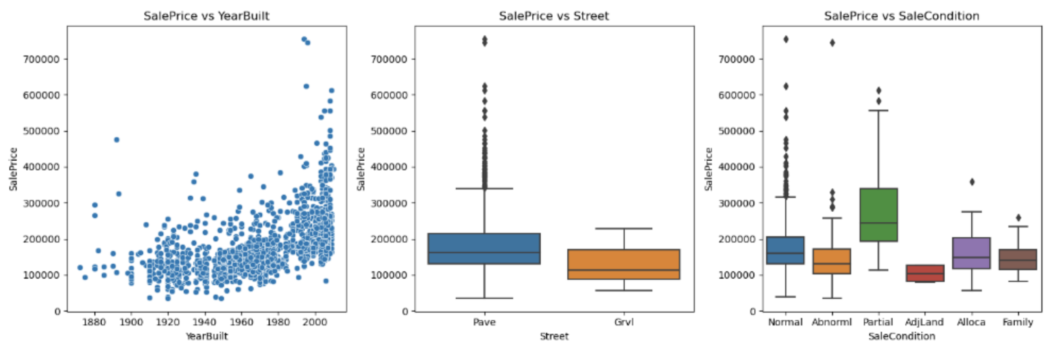
```python
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
sns.scatterplot(data=df, x='YearBuilt', y='SalePrice')
plt.title('SalePrice vs YearBuilt')

plt.subplot(1, 3, 2)
sns.boxplot(data=df, x='Street', y='SalePrice')
plt.title('SalePrice vs Street')

plt.subplot(1, 3, 3)
sns.boxplot(data=df, x='SaleCondition', y='SalePrice')
plt.title('SalePrice vs SaleCondition')

plt.tight_layout()
plt.show()
```



Now I will define a new variable called non-living space area, which combine pool, garage, and WoodDecks. This variable should tell if people are willing to pay a higher price for larger nonliving leisure areas.

```python
In [44]: df['NonlivingArea'] = df['GarageArea'] + df['PoolArea'] + df['WoodDeckSF']
         # Display the first few rows to verify the new predictor
         print(df['NonlivingArea'].head())

0     548
1     758
2     608
3     642
4    1028
Name: NonlivingArea, dtype: int64
```

Now I will use two methods to do minmascaler/standard scale. First I can use sklearn's scaler functions. This will scale down Saleprice based on their minmax difference or standard deviation.

```python
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler

sale_price = df[['SalePrice']]

# Min-Max Scaling
min_max_scaler = MinMaxScaler()
sale_price_minmax_scaled = min_max_scaler.fit_transform(sale_price)

# Standard Scaling (Z-score normalization)
standard_scaler = StandardScaler()
sale_price_standard_scaled = standard_scaler.fit_transform(sale_price)

# Convert scaled arrays back to DataFrame for better visualization
sale_price_minmax_scaled_df = pd.DataFrame(sale_price_minmax_scaled, columns=['SalePrice_MinMaxScaled'])
sale_price_standard_scaled_df = pd.DataFrame(sale_price_standard_scaled, columns=['SalePrice_StandardScaled'])

# Concatenate original and scaled dataframes for comparison
scaled_df = pd.concat([sale_price, sale_price_minmax_scaled_df, sale_price_standard_scaled_df], axis=1)

# Display the first few rows of the scaled dataframe
print(scaled_df.head())
```

```
   SalePrice  SalePrice_MinMaxScaled  SalePrice_StandardScaled
0     208500                0.241078                  0.347273
1     181500                0.203583                  0.007288
2     223500                0.261908                  0.536154
3     140000                0.145952                 -0.515281
4     250000                0.298709                  0.869843
```

I can also use Math calculation to achieve the same result.

```python
# Extract the dependent variable
sale_price = df['SalePrice']

# Min-Max Scaling
min_sale_price = sale_price.min()
max_sale_price = sale_price.max()
scaled_minmax = (sale_price - min_sale_price) / (max_sale_price - min_sale_price)

# Standard Scaling (Z-score normalization)
mean_sale_price = sale_price.mean()
std_sale_price = sale_price.std()
scaled_standard = (sale_price - mean_sale_price) / std_sale_price

# Convert scaled arrays back to DataFrame for better visualization
scaled_minmax_df = pd.DataFrame(scaled_minmax)
scaled_standard_df = pd.DataFrame(scaled_standard)

# Concatenate original and scaled dataframes for comparison
scaled_df = pd.concat([sale_price, scaled_minmax_df, scaled_standard_df], axis=1)

# Display the first few rows of the scaled dataframe
print(scaled_df.head())
```

```
   SalePrice  SalePrice  SalePrice
0     208500   0.241078   0.347154
1     181500   0.203583   0.007286
2     223500   0.261908   0.535970
3     140000   0.145952  -0.515105
4     250000   0.298709   0.869545
```