Fourth Industrial Summer School

Day 2: Data Analysis Foundations - Morning

Exploratory Data Analysis

Session Objectives

- ✓ Seaborn
  - heatmap
  - Correlation matrix
- ✓ Checking missing data
- ✓ Removing features with missing data
- ✓ Exploring
  - numeric data
  - categorical data

Exploratory Data Analysis (EDA)

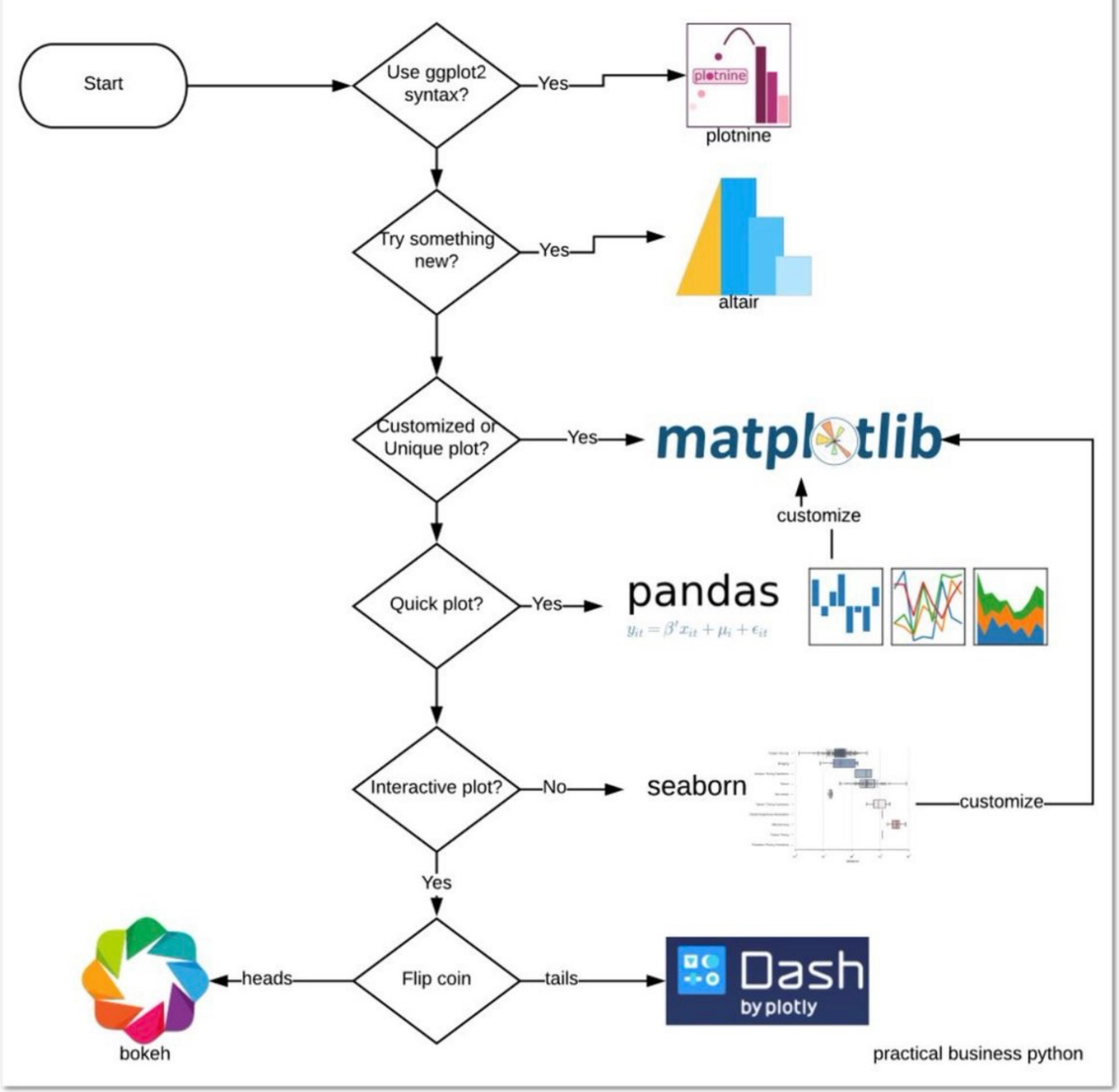
- To give insight into your data
- Understand the underlying structure
- Extract important features and relationships
- Generate Hypothesis

Seaborn

- Seaborn is a Python data visualization library based on matplotlib.
- It provides a high-level interface for drawing attractive and informative statistical graphics.
- It introduces additional plot types.
- It also makes your traditional Matplotlib plots look a bit prettier.
- Similar (in style) to the popular ggplot2 library in R

Link: https://seaborn.pydata.org/

Choosing a python visualization tool



Data Exploration example

- Employee2.csv
  - Company employees information
  - Used before as Employee.csv, but with some missing data

emp_id	Gender	Age	Sales	BMI	Income
1	M	34	123	Normal	350
2	F	40	114	Overweight	450
3	F	37	135	Obesity	169
4	M	30	139	Underweight	
5	F	44	117	Underweight	183
6	M	36	121	Normal	80
7	M		133	Obesity	166
8	F	26	140	Normal	120
9		32	133	Normal	75
10	M	36	133	Underweight	40

Import

- Import libraries

```python
import pandas as pd
import seaborn as sns
sns.set()
import matplotlib.pyplot as plt
```

- Notice the missing data

```python
df = pd.read_csv('Employee2.csv')
df.head()
```

	emp_id	Gender	Age	Sales	BMI	Income
0	1	M	34.0	123	Normal	350.0
1	2	F	40.0	114	Overweight	450.0
2	3	F	37.0	135	Obesity	169.0
3	4	M	30.0	139	Underweight	NaN
4	5	F	44.0	117	Underweight	183.0

Data Shape

- Data shape
  - 10 rows
  - 6 columns

```
df.shape
```

```
(10, 6)
```

- Data information
  - Check data types

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
emp_id     10 non-null int64
Gender     9 non-null object
Age        9 non-null float64
Sales      10 non-null int64
BMI        10 non-null object
Income     9 non-null float64
dtypes: float64(2), int64(2), object(2)
memory usage: 560.0+ bytes
```

Descriptive Statistics

- Specific variable statistics

```
df.Income.describe()
```

```
count      9.000000
mean     181.444444
std      135.246175
min       40.000000
25%       80.000000
50%      166.000000
75%      183.000000
max      450.000000
Name: Income, dtype: float64
```

Missing Values

- Check missing values

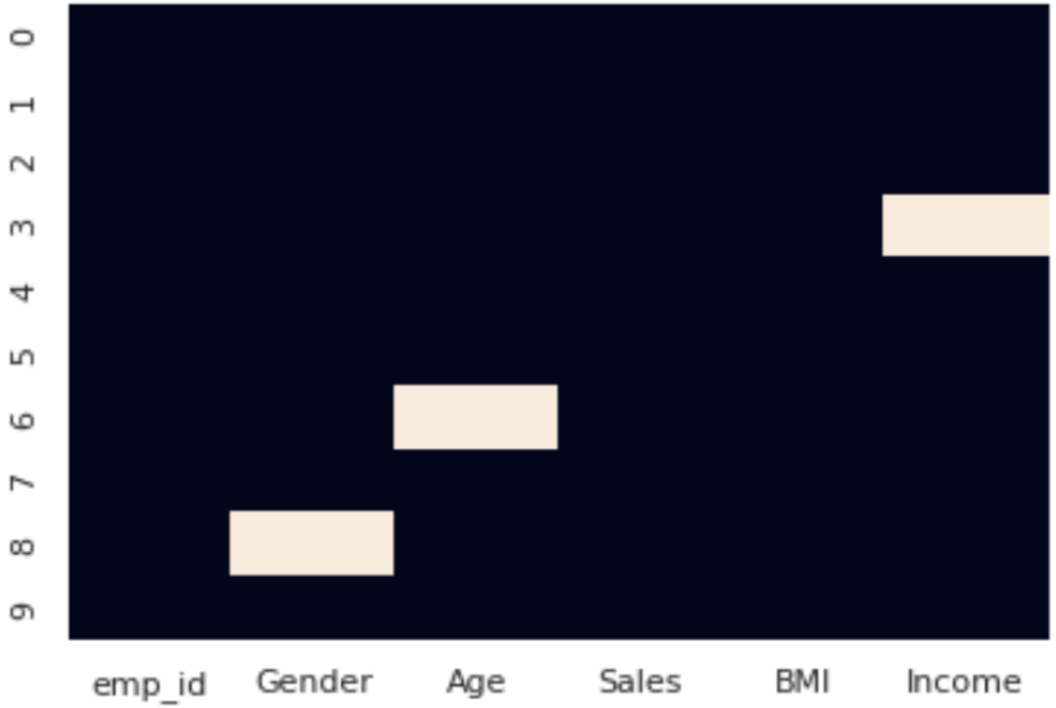
```
1  df.isnull().sum()
```

```
emp_id    0
Gender    1
Age       1
Sales     0
BMI       0
Income    1
dtype: int64
```

Visualize Missing Values

- Use heatmap to visualize the missing values

```
1   sns.heatmap(df.isnull(), cbar=False)
```



Link: https://seaborn.pydata.org/generated/seaborn.heatmap.html
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.isnull.html

Handle Missing Data

- Fill missing **numerical** values with mean

```
df_num = df.select_dtypes(include=['number'])
df[df_num.columns] = df_num.fillna(df_num.mean())
```

- Fill missing **categorical** values with mode

```
df_cat = df.select_dtypes(exclude=['number'])
mode_values = df_cat.mode().iloc[0]
df[df_cat.columns] = df_cat.fillna(mode_values)
```

Re-Visualize Missing Values

- Replot the heatmap

```
df = pd.concat([df_num, df_cat], axis=1)
sns.heatmap(df.isnull(), cbar=False)
```

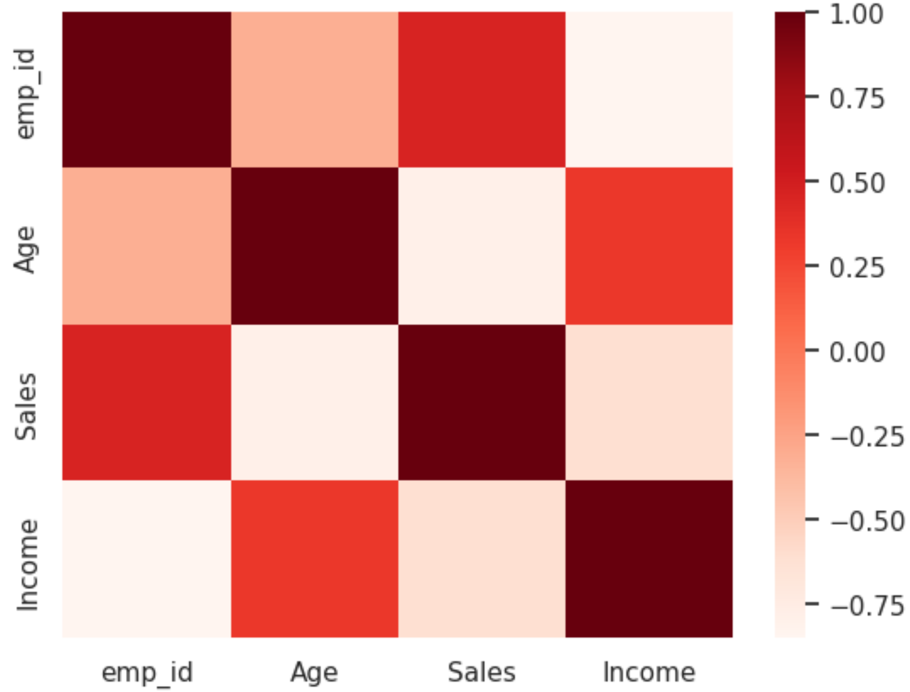
<Axes: >

Link: h
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.isnull.html

Correlation Matrix

- Correlation as a heatmap

```
1    sns.heatmap(df_num.corr(),cmap='Reds')
```

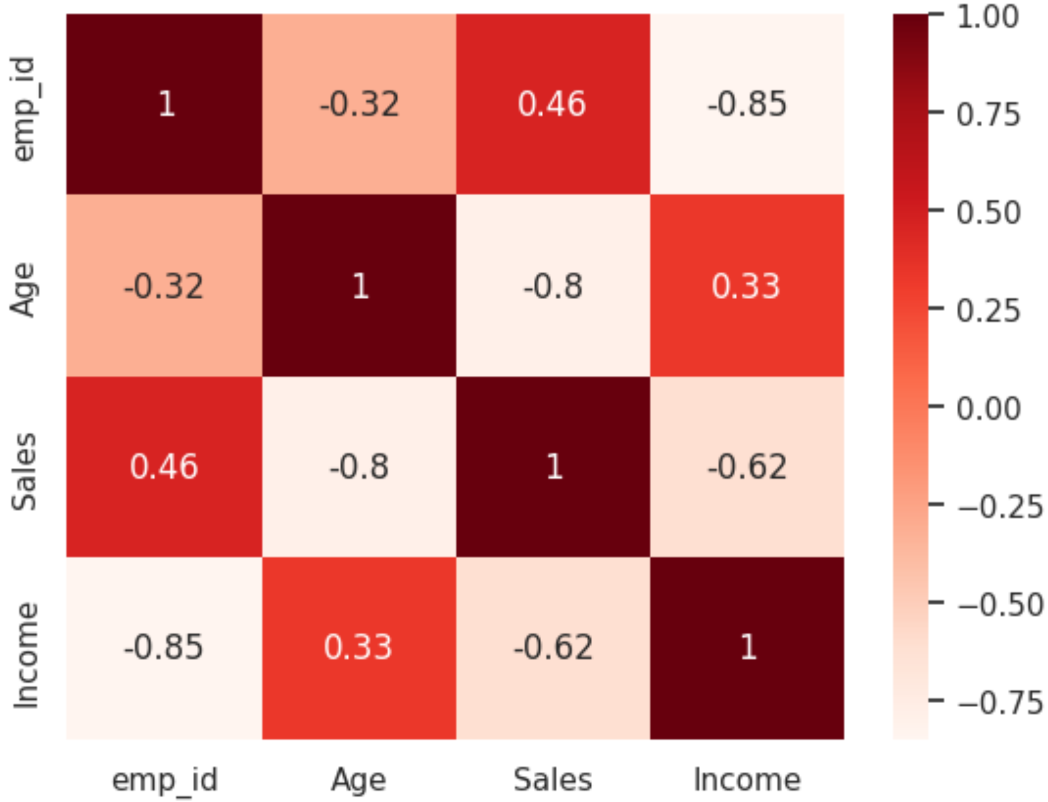


Link: https://seaborn.pydata.org/generated/seaborn.heatmap.html
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html

Correlation Matrix

- Correlation as a heatmap with correlation annotation

```
1   sns.heatmap(df_num.corr(), annot=True, cmap='Reds')
```

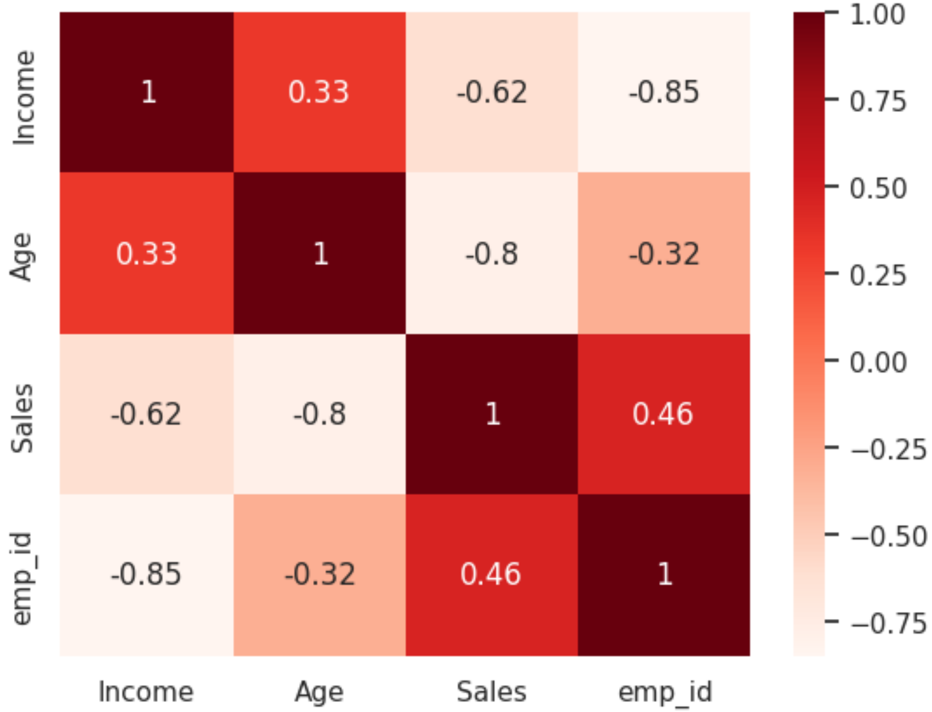


Link: https://seaborn.pydata.org/generated/seaborn.heatmap.html
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html

Correlation Matrix

- Correlation matrix **sorted**
  - Calculate the top k correlations with 'Income'

```
k = len(df)
cols = df_num.corr().nlargest(k, 'Income')['Income'].index
cm = df_num[cols].corr()
sns.heatmap(cm, annot=True, cmap='Reds')
```



Link: https://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.DataFrame.nlargest.html

Data Exploration session

House Sales

House Sales Data Exploration

- HouseSales.csv

```
df = pd.read_csv('HousePrices.csv')
df.head()
```

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

5 rows × 81 columns

House Sales Data Exploration

- Data information

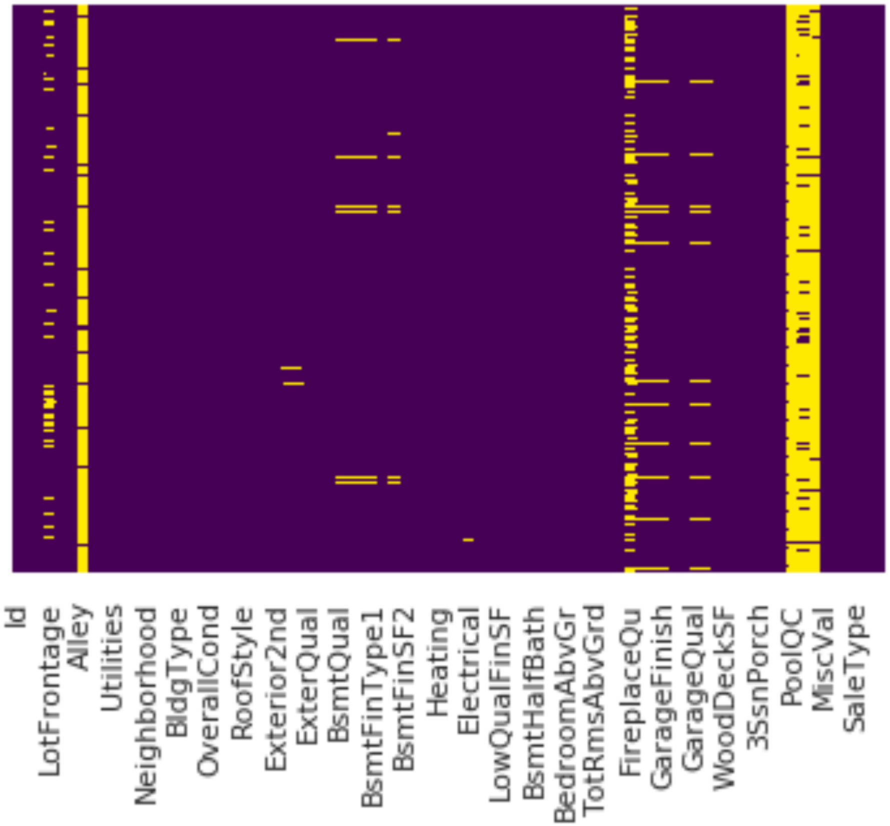
```
1   df.info()
```

```
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
Id                 1460 non-null int64
MSSubClass         1460 non-null int64
MSZoning           1460 non-null object
LotFrontage        1201 non-null float64
LotArea            1460 non-null int64
Street             1460 non-null object
Alley              91 non-null object
LotShape           1460 non-null object
LandContour        1460 non-null object
Utilities          1460 non-null object
LotConfig          1460 non-null object
LandSlope          1460 non-null object
Neighborhood       1460 non-null object
Condition1         1460 non-null object
Condition2         1460 non-null object
BldgType           1460 non-null object
HouseStyle         1460 non-null object
```

House Sales Data Exploration

- Investigate missing data
  - Yellow color

```
1   sns.heatmap(df.isnull(), cbar=False, yticklabels=False, cmap='viridis')
```

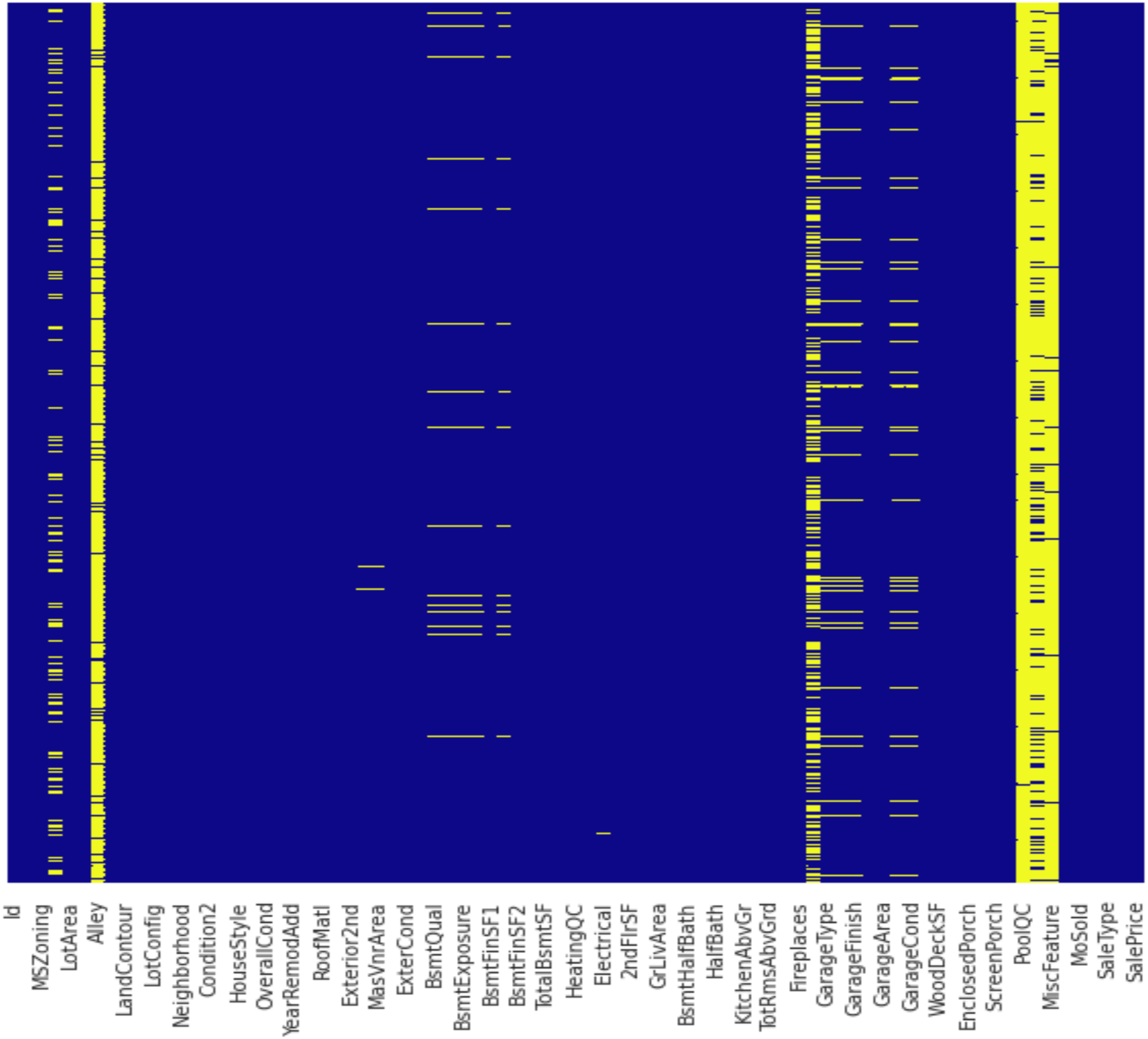


House Sales Data Exploration

- Increase the figure size for better quality
- Try different colors

```
plt.subplots(figsize=(10,10))
sns.heatmap(df.isnull(), cbar=False, yticklabels=False, cmap='plasma')
```

https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.subplots.html


Id
MSZoning
LotArea
Alley
LandContour
LotConfig
Neighborhood
Condition2
HouseStyle
OverallCond
YearRemodAdd
RoofMatl
Exterior2nd
MasVnrArea
ExterCond
BsmtQual
BsmtExposure
BsmtFinSF1
BsmtFinSF2
TotalBsmtSF
HeatingQC
Electrical
2ndFlrSF
GrLivArea
BsmtHalfBath
HalfBath
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageType
GarageFinish
GarageArea
GarageCond
WoodDeckSF
EnclosedPorch
ScreenPorch
PoolQC
MiscFeature
MoSold
SaleType
SalePrice

House Sales Data Exploration

- What are your observations?
- Some features has many missing values
  - Should we keep them?
- Let us remove features with too many missing data

House Sales Data Exploration

- Remove features with 50% or less missing data

```
df2 = df[[column for column in df if df[column].count() / len(df) >= 0.5]]
del df2['Id']
print("Features dropped:")
for c in df.columns:
    if c not in df2.columns:
        print(c)
df = df2
```

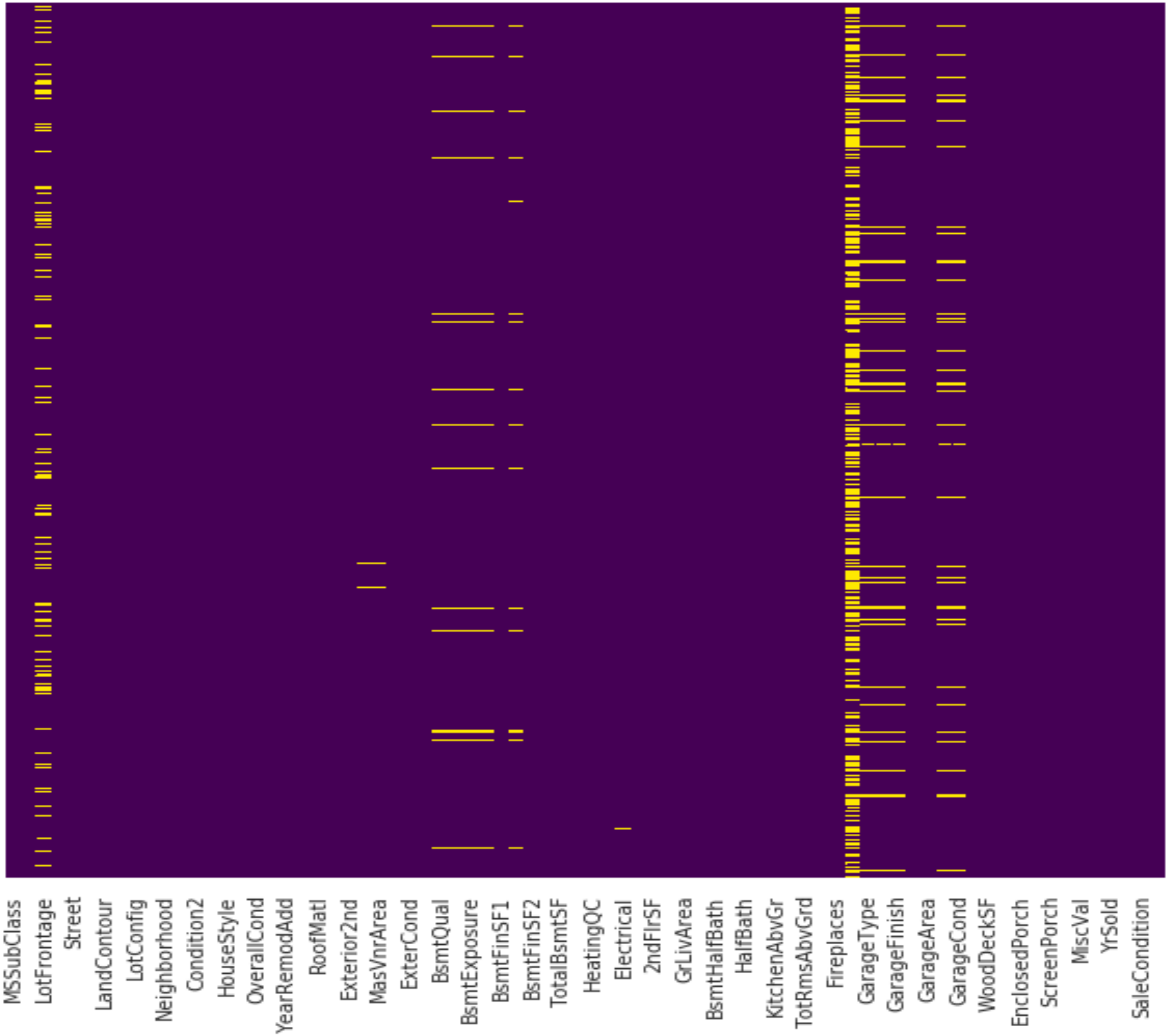
- Code to check which features were dropped

```
Features dropped:
Id
Alley
PoolQC
Fence
MiscFeature
```

House Sales Data Exploration

- Replot heatmap to check missing values

```
plt.subplots(figsize=(10,10))
sns.heatmap(df.isnull(), cbar=False, yticklabels=False, cmap='viridis')
```


MSSubClass
LotFrontage
Street
LandContour
LotConfig
Neighborhood
Condition2
HouseStyle
OverallCond
YearRemodAdd
RoofMatl
Exterior2nd
MasVnrArea
ExterCond
BsmtQual
BsmtExposure
BsmtFinSF1
BsmtFinSF2
TotalBsmtSF
HeatingQC
Electrical
2ndFlrSF
GrLivArea
BsmtHalfBath
HalfBath
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageType
GarageFinish
GarageArea
GarageCond
WoodDeckSF
EnclosedPorch
ScreenPorch
MiscVal
YrSold
SaleCondition

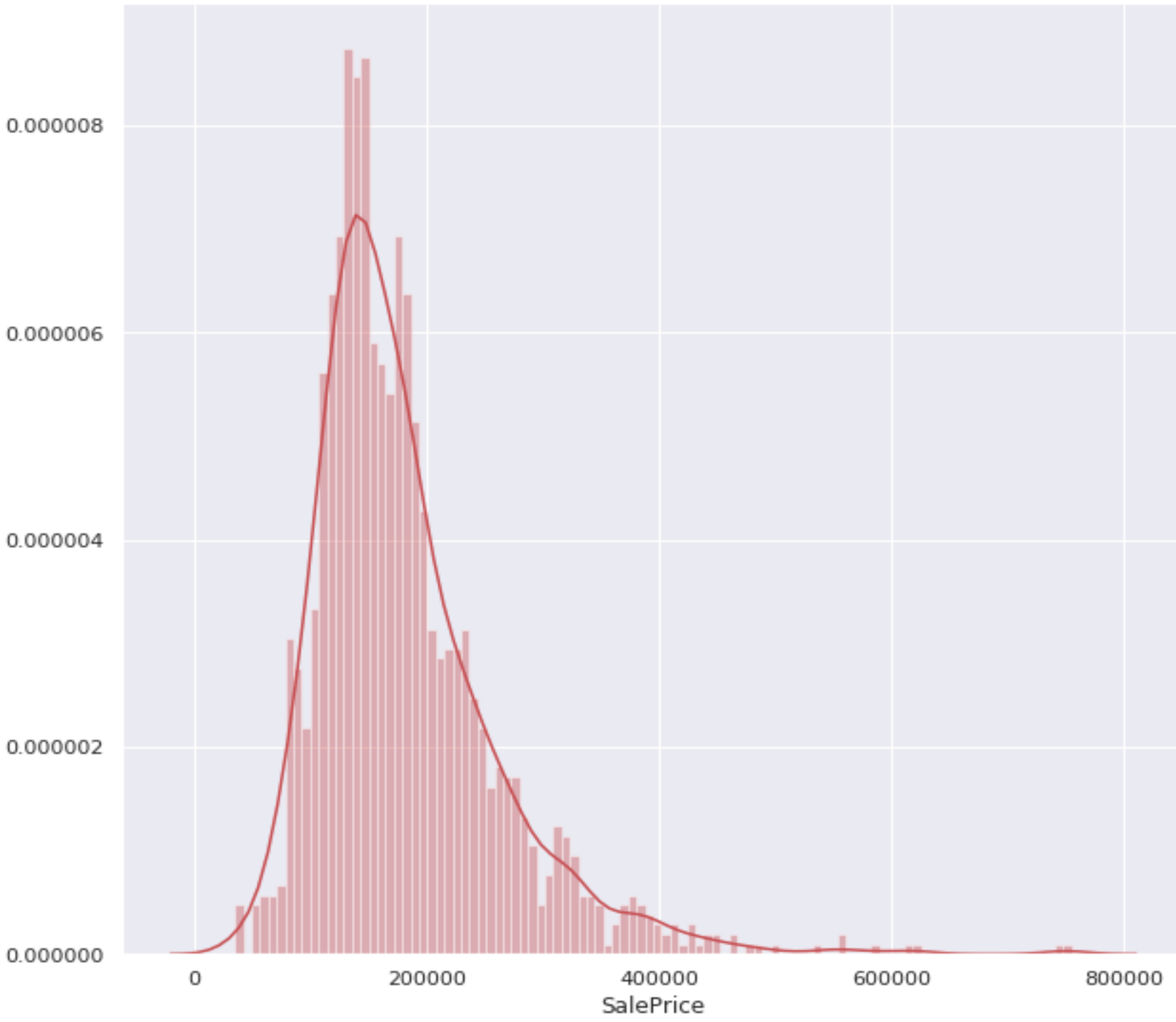
House Sales Data Exploration

- Study the House sales distribution
  - Seaborn provides a nice histogram plot function
    - displot

```
print(df['SalePrice'].describe())
plt.figure(figsize=(10, 10))
sns.distplot(df['SalePrice'], color='r', bins=100)
```

```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

Link: https://seaborn.pydata.org/generated/seaborn.distplot.html


0.000008
0.000006
0.000004
0.000002
0.000000
0
200000
400000
600000
800000
SalePrice

House Sales Data Exploration

- Data contains both Numerical and Categorical data
- We will explore both types in the next slides

House Sales Data Exploration

- Numerical data exploration

- First, we need to filter the data based on their type
  - int and float

```
df_num = df.select_dtypes(include = ['float64', 'int64'])
```

  - New dataframe with only numeric type

  - Check how many features were retrieved.

House Sales Data Exploration

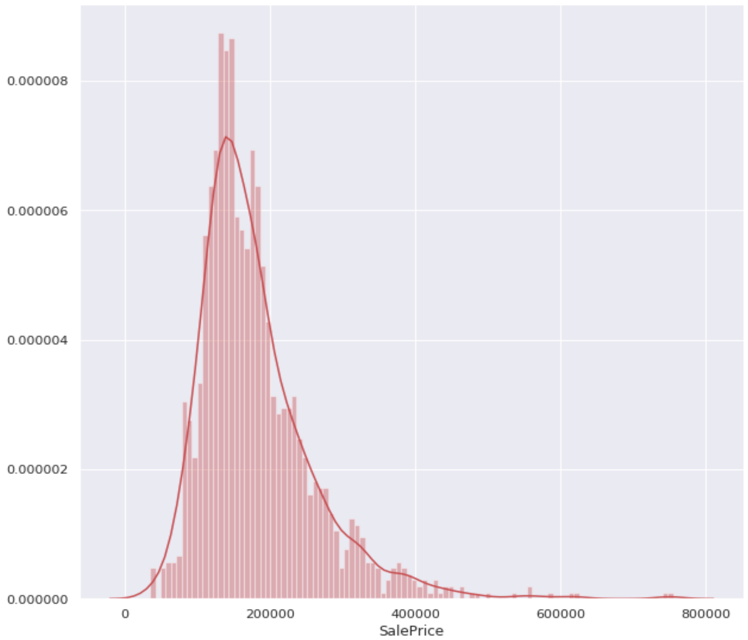
- Explore all the features using histograms

```
df_num.hist(figsize=(20, 20), bins=50, xlabelsize=8, ylabelsize=8)
```


1stFlrSF
2ndFlrSF
3SsnPorch
BedroomAbvGr
BsmtFinSF1
BsmtFinSF2
BsmtFullBath
BsmtHalfBath
BsmtUnfSF
EnclosedPorch
Fireplaces
FullBath
GarageArea
GarageCars
GarageYrBlt
GrLivArea
HalfBath
KitchenAbvGr
LotArea
LotFrontage
LowQualFinSF
MSSubClass
MasVnrArea
MiscVal
MoSold
OpenPorchSF
OverallCond
OverallQual
PoolArea
SalePrice
ScreenPorch
TotRmsAbvGrd
TotalBsmtSF
WoodDeckSF
YearBuilt
YearRemodAdd
YrSold

House Sales Data Exploration

- Which features histograms shares similar distribution as Sales ??
  - e.g. 1stFlrSF feature



House Sales Data Exploration

- Next study correlation between features and Sales Price
  - Focus on high correlation with > 0.5 score

```
df_num_corr = df_num.corr()['SalePrice']
selectedFeatures=df_num_corr[abs(df_num_corr)>0.5].sort_values(ascending=False)
print(selectedFeatures)
```

```
SalePrice       1.000000
OverallQual     0.790982
GrLivArea       0.708624
GarageCars      0.640409
GarageArea      0.623431
TotalBsmtSF     0.613581
1stFlrSF        0.605852
FullBath        0.560664
TotRmsAbvGrd    0.533723
YearBuilt       0.522897
YearRemodAdd    0.507101
Name: SalePrice, dtype: float64
```

- Do you think correlation is effected by outliers?
  - Make a scatter plot to observe the relationships

<u>Break</u>

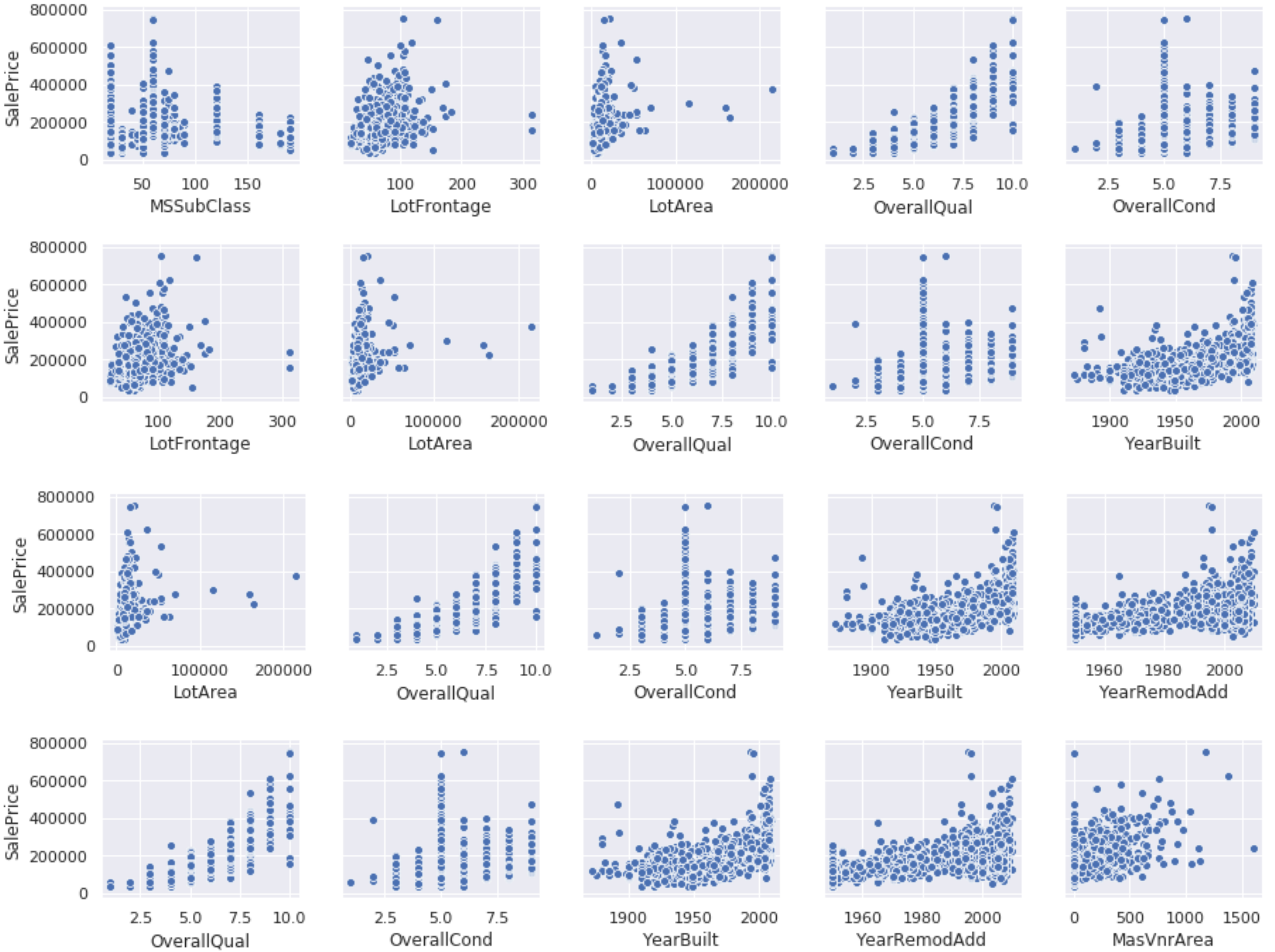
House Sales Data Exploration

- Plot all features scatter plot
  - pairplot

```
for i in range(0,len(df_num.columns),5):
    sns.pairplot(data=df_num,x_vars=df_num.columns[i:i+5], y_vars=['SalePrice'])
```

- check if our correlated values have a **linear relationship** to the Sale Price.

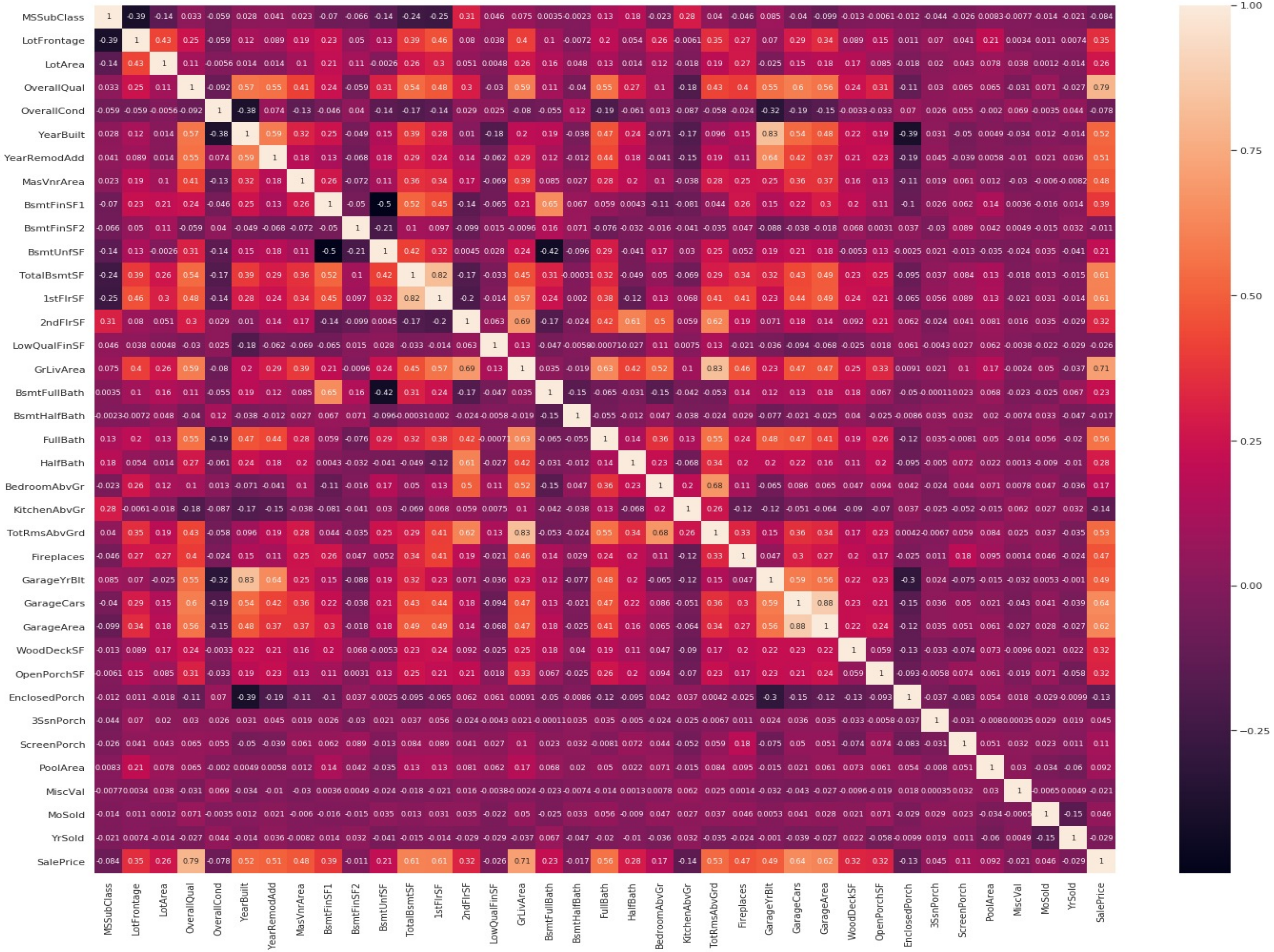
Link: https://seaborn.pydata.org/generated/seaborn.pairplot.html


SalePrice
MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
SalePrice
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
SalePrice
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
SalePrice
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea

House Sales Data Exploration

- Plotting all the numerical features in a seaborn pairplot will be hard to interpret.
  - **Heatmap**

```
corr = df_num.corr()
plt.figure(figsize=(24,20))
sns.heatmap(corr, annot=True)
```


MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice
1.00
0.75
0.50
0.25
0.00
−0.25

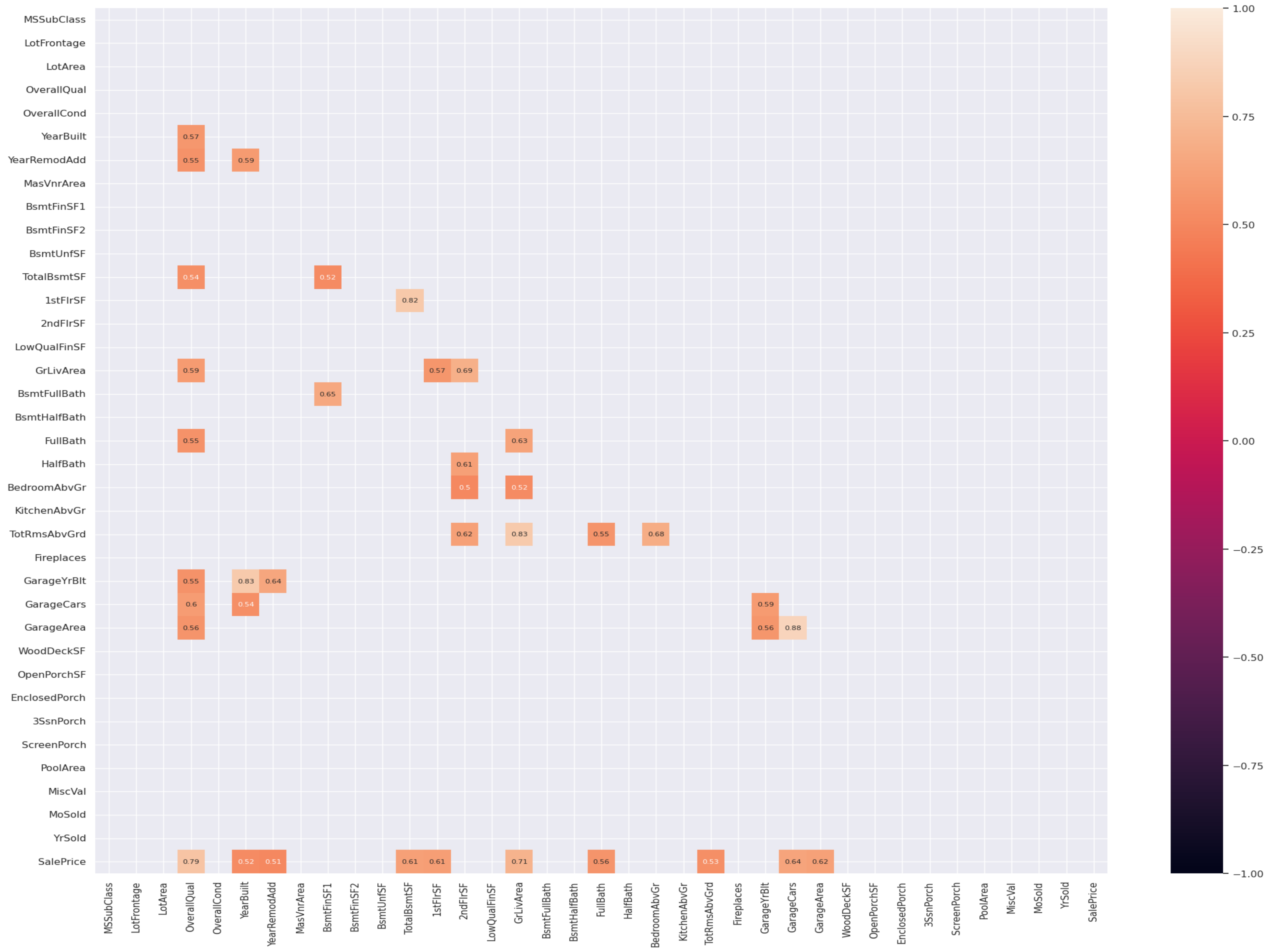
House Sales Data Exploration

- Even this heatmap is difficult to interpret

- Some optimization will help!!

```
corr = df_num.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
plt.figure(figsize=(24, 20))
sns.heatmap(corr[(corr>=0.5)|(corr<=-0.5)],
            mask=mask,
            annot=True,
            annot_kws={'size':8},
            vmin=-1,
            vmax=1)
```

- See the heatmap and state your observations?
  - There is a **strong negative** correlation between BsmtUnfSF and BsmtFinSF2
  - **Others?**



MSSubClass
LotFrontage
LotArea
OverallQual
OverallCond
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
TotalBsmtSF
1stFlrSF
2ndFlrSF
LowQualFinSF
GrLivArea
BsmtFullBath
BsmtHalfBath
FullBath
HalfBath
BedroomAbvGr
KitchenAbvGr
TotRmsAbvGrd
Fireplaces
GarageYrBlt
GarageCars
GarageArea
WoodDeckSF
OpenPorchSF
EnclosedPorch
3SsnPorch
ScreenPorch
PoolArea
MiscVal
MoSold
YrSold
SalePrice
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

House Sales Data Exploration

- Can we decrease the number of features based on our previous observations?

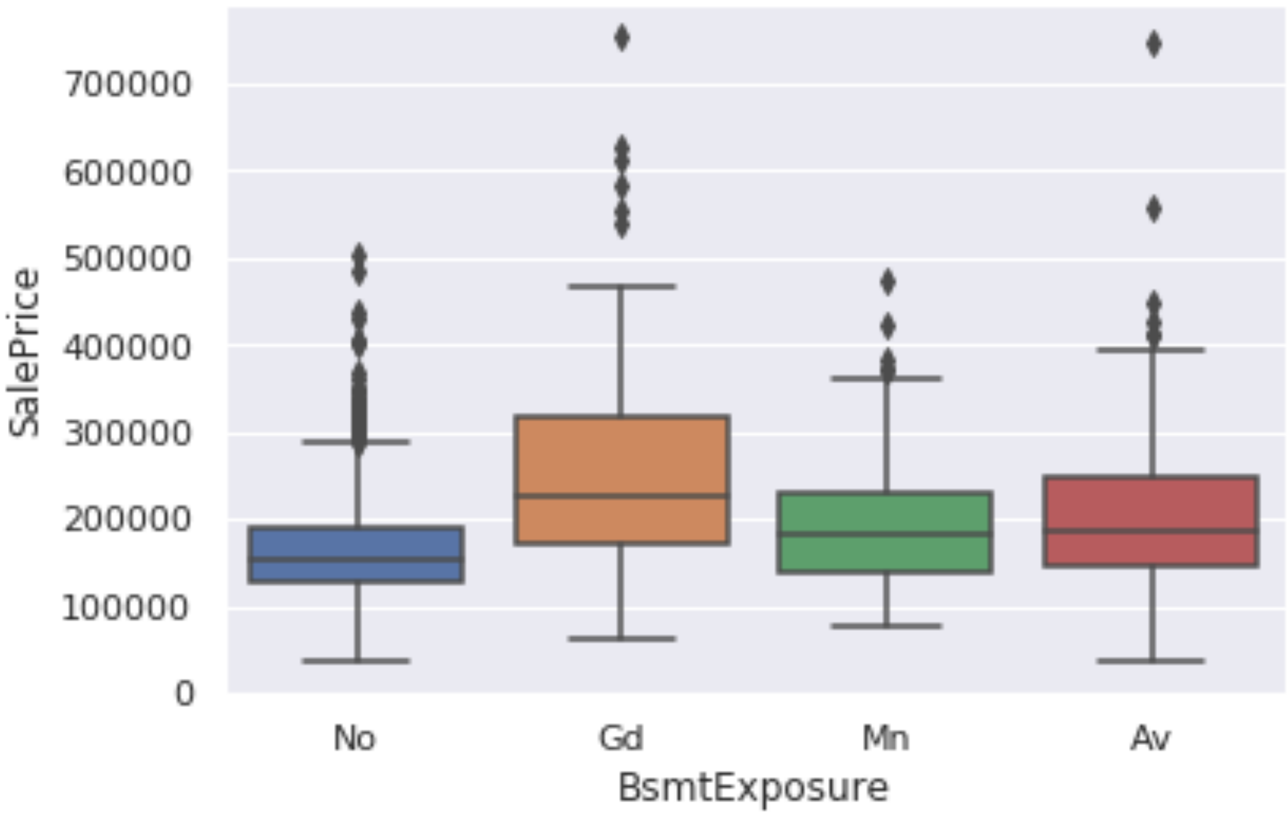
House Sales Data Exploration

- Let us look at some **categorical** features of our dataset and see if we can find some insight in them.
  - Consider (BsmtExposure) feature

- Use boxplot

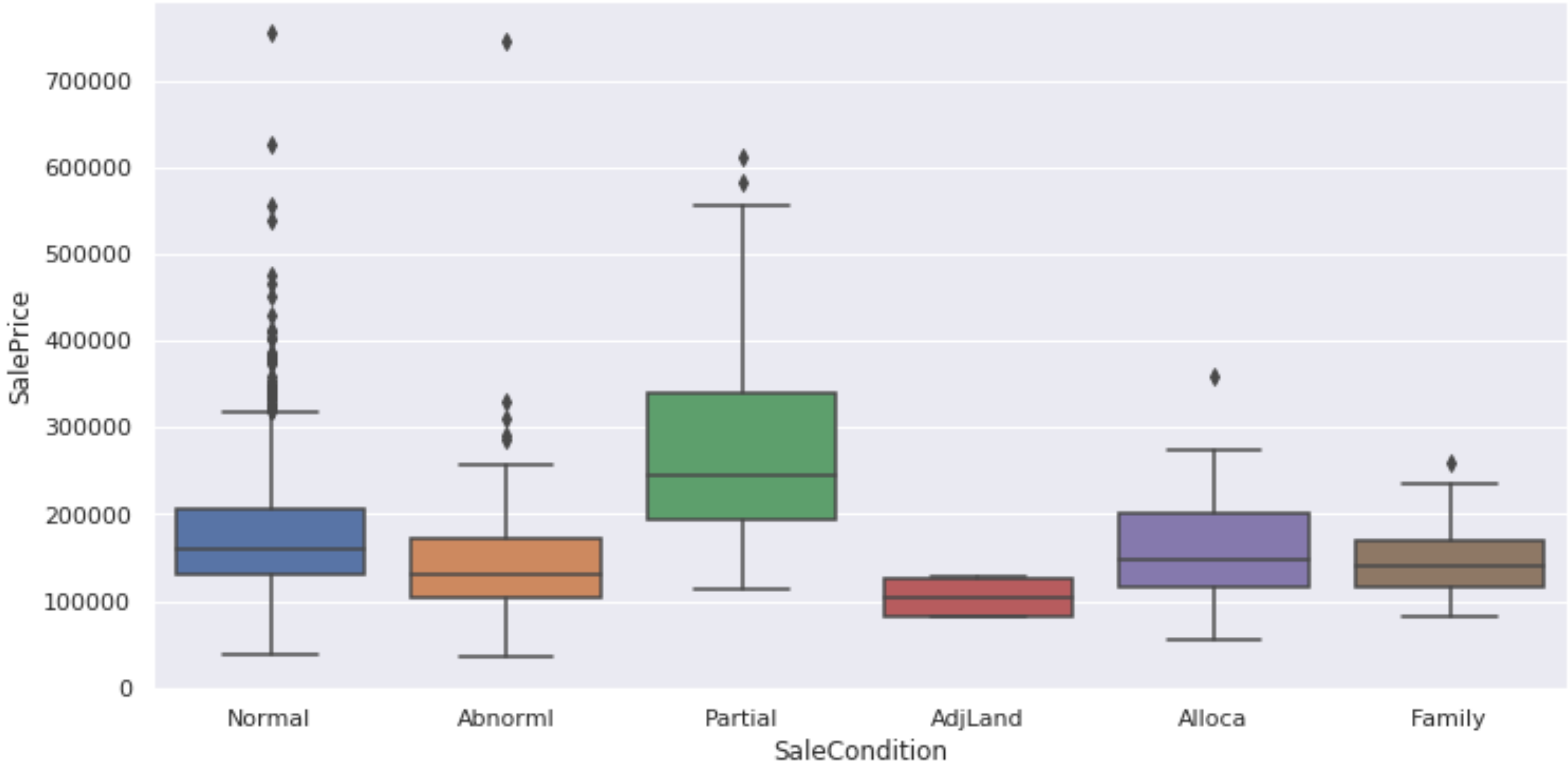
```
sns.boxplot(x='BsmtExposure', y='SalePrice', data=df)
```

House Sales Data Exploration



House Sales Data Exploration

- Check Sales Condition feature?



Exercise

Iris Dataset

Iris dataset

- Introduced by the British statistician / biologist Ronald Fisher in his 1936

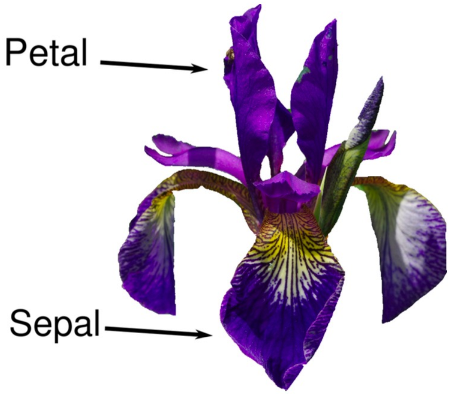
- It is a multi-class classification problem.

Number of Instances:	150	Area:	Life
Number of Attributes:	4	Date Donated	1988-07-01
Missing Values?	No	Number of Web Hits:	4457610

https://archive.ics.uci.edu/ml/datasets/Iris

Iris dataset (cont'd)

- Balanced dataset
- 150 observations
- 4 numeric variables
  - Sepal length in cm
  - Sepal width in cm
  - Petal length in cm
  - Petal width in cm
- 1 target variable
  - **Class** (Iris Setosa, Iris Versicolour, Iris Virginica)



Iris dataset (cont'd)

- No need to download it.

https://scikit-learn.org/stable/datasets/toy_dataset.html

`load_boston`(*[, return_X_y])	DEPRECATED: `load_boston` is deprecated in 1.0 and will be removed in 1.2.
`load_iris`(* [, return_X_y, as_frame])	Load and return the iris dataset (classification).
`load_diabetes`(* [, return_X_y, as_frame])	Load and return the diabetes dataset (regression).
`load_digits`(* [, n_class, return_X_y, as_frame])	Load and return the digits dataset (classification).
`load_linnerud`(* [, return_X_y, as_frame])	Load and return the physical exercise Linnerud dataset.
`load_wine`(* [, return_X_y, as_frame])	Load and return the wine dataset (classification).
`load_breast_cancer`(* [, return_X_y, as_frame])	Load and return the breast cancer wisconsin dataset (classification).

Iris dataset (cont'd)

```
from sklearn.datasets import load_iris
iris_dataset = load_iris()
```

return a **Dictionary** like object

Returns

data : `~sklearn.utils.Bunch`
Dictionary-like object, with the following attributes.

```
import seaborn as sns
iris = sns.load_dataset('iris')
```

return a **DataFrame**

Returns

df : `pandas.DataFrame`
Tabular data, possibly with some preprocessing applied.

```
sns.load_dataset("iris")
```

Iris data Vs. Real data

Exercise

Perform EDA on Iris dataset by doing:

- Descriptive statistics
- Removing duplicate data entries
- Compare between various species based on petal length and width.
  - Write your data insights in a notebook text
- Boxplot petal width distribution over species
  - Write your data insights in a notebook text