

第3回 対面授業（今回からピッチあげるよ。頑張ってくださいね）

課題1: データ前処理の実習

機械学習を実施するにあたり「データを作成（調整）」することは非常に重要です。予測の精度にも関わります。機械学習を始める前にデータを調整することを **前処理** と言います。データの前処理は、データ分析の全行程のうち約8割を占める重要な工程です。

ここではデータの前処理について理解いただくための課題を用意しました。この課題ではkaggleのcompetitionで使用されたデータを用いてデータの前処理方法を確認します。

各列の詳細は下記サイトをご参照ください。 <https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data> (<https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data>)

各セルに入っているコメントの下に、実行するコードを記入してください。わからない場合は、各種ライブラリの公式ドキュメントを参照しましょう。

1. 必要なモジュールの読み込み

In [2]:

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

matplotlib.style.use('ggplot')

%matplotlib inline
```

2. データの読み込み

CSVファイル"kaggle_housing_price.csv"を読み込み、内容を確認します。

In [3]:



```
# データを変数datasetに読み込む
housing_data = pd.read_csv("kaggle_housing_price.csv")
housing_data
```

Out[3]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandConto
0	1	60	RL	65.0	8450	Pave	NaN	Reg	L
1	2	20	RL	80.0	9600	Pave	NaN	Reg	L
2	3	60	RL	68.0	11250	Pave	NaN	IR1	L
3	4	70	RL	60.0	9550	Pave	NaN	IR1	L
4	5	60	RL	84.0	14260	Pave	NaN	IR1	L
...
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	L
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	L
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	L
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	L
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	L

1460 rows × 81 columns

In [4]:



```
# データを最初の5行だけ表示
housing_data.head(5)
```

Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
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	/

5 rows × 81 columns

演習：データを最後の5行だけ表示してください

In [6]:



```
# データを最後の5行だけ表示
housing_data.tail()

# データを最後の10行だけ表示
housing_data.tail(10)
```

Out[6]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandConto
1450	1451	90	RL	60.0	9000	Pave	NaN	Reg	L
1451	1452	20	RL	78.0	9262	Pave	NaN	Reg	L
1452	1453	180	RM	35.0	3675	Pave	NaN	Reg	L
1453	1454	20	RL	90.0	17217	Pave	NaN	Reg	L
1454	1455	20	FV	62.0	7500	Pave	Pave	Reg	L
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	L
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	L
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	L
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	L
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	L

10 rows × 81 columns

DataFrameの shape プロパティで全データの行数と列数を取得できます。

参照：<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.shape.html>
[\(http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.shape.html\)](http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.shape.html)

In [7]:



```
# データの行数、列数を表示
housing_data.shape
```

Out[7]:

(1460, 81)

3. 要約統計量を出力する

データ数、平均や中央値、標準偏差などの統計量を確認し、データへの理解を深めます。

なお、DataFrameの describe() を使うと、様々な統計量の情報を要約として表示してくれます。

参考：<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.describe.html>
[\(http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.describe.html\)](http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.describe.html)

In [8]:



```
# 要約統計量を表示  
housing_data.describe()
```

Out[8]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.111111
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.111111
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000

8 rows × 8 columns

4. 基本的なデータの操作

データの抽出・列の分割について学びます。

インデックスを用いた行の指定

In [9]:

```
# インデックス番号0~10の行を抽出
housing_data.iloc[0:10,0:81]
```

Out[9]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	U
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
5	6	50	RL	85.0	14115	Pave	NaN	IR1		Lvl
6	7	20	RL	75.0	10084	Pave	NaN	Reg		Lvl
7	8	60	RL	NaN	10382	Pave	NaN	IR1		Lvl
8	9	50	RM	51.0	6120	Pave	NaN	Reg		Lvl
9	10	190	RL	50.0	7420	Pave	NaN	Reg		Lvl

10 rows × 81 columns

In [10]:

```
# "SalePrice"と"LotArea"の列を抽出し、最初の10行だけ表示
housing_data.loc[0:9, ["SalePrice","LotArea"]]
```

Out[10]:

	SalePrice	LotArea
0	208500	8450
1	181500	9600
2	223500	11250
3	140000	9550
4	250000	14260
5	143000	14115
6	307000	10084
7	200000	10382
8	129900	6120
9	118000	7420

カラム間の演算

PandasのDataFrameでは、列同士の四則演算、ならびに新しい列を追加することができます。

たとえば、df というDataFrameの変数があり、中に a , b というカラムが存在するとき、df['c'] = df['a'] + df['b'] と記述することで、a列とb列の加算結果を c という新しい列として df に追加してくれます。

In [11]:

```
# "1stFlrSF"と"2ndFlrSF"を合計した"FlrSF_total"を新たな列としてdatasetに加える
df=pd.DataFrame(housing_data)
df['FlrSF_total'] = df['1stFlrSF'] + df['2ndFlrSF']
```

In [12]:

```
# datasetからiloc関数を使用して"1stFlrSF"と"2ndFlrSF"、"FlrSF_total"を先頭から5行分表示し、正しく追加
df.loc[0:4, ["1stFlrSF", "2ndFlrSF", "FlrSF_total"]]
```

Out[12]:

	1stFlrSF	2ndFlrSF	FlrSF_total
0	856	854	1710
1	1262	0	1262
2	920	866	1786
3	961	756	1717
4	1145	1053	2198

ダミー変数の作成

列 SaleType は WD , New , COD など、合わせて9種類の値しかありません。このようなデータをもつ列は一般的に **カテゴリ変数** と呼んでいます（Lesson6以降で改めて説明します）。カテゴリ変数のデータは、**ダミー変数** にしてあげましょう。たとえば SaleType ですと、

- SaleType_WD : SaleType のデータが WD なら1、それ以外の場合は0が入る列
- SaleType_New : SaleType のデータが New なら1、それ以外の場合は0が入る列
- SaleType_COD : SaleType のデータが COD なら1、それ以外の場合は0が入る列

(以下省略)

このようにして、SaleType のデータの種類の数に合わせて9個の新しい列を作成します。この9個の列には、1つが 1 で、残りは全て 0 が入っています。これがダミー変数です。

ダミー変数化するのに最も楽な方法は、Pandasの get_dummies() を使う方法です。

参考：http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.get_dummies.html
[\(http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.get_dummies.html\)](http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.get_dummies.html)

In [13]:



```
# 列'SaleType'をダミー変数に展開したものを変数 dataset に上書きします。
dataset=pd.get_dummies(df['SaleType'])
print(dataset)
```

	COD	CWD	Con	ConLD	ConLI	ConLw	New	Oth	WD
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	1
...
1455	0	0	0	0	0	0	0	0	1
1456	0	0	0	0	0	0	0	0	1
1457	0	0	0	0	0	0	0	0	1
1458	0	0	0	0	0	0	0	0	1
1459	0	0	0	0	0	0	0	0	1

[1460 rows x 9 columns]

In [14]:



```
# ダミー変数が作成されていることを確認します (datasetの最初の5行だけ出力)
dataset.head()
```

Out[14]:

	COD	CWD	Con	ConLD	ConLI	ConLw	New	Oth	WD
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1
3	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	1

フィルタリング

特定の条件の行のみ抽出することができます。そのために、DataFrameの `query()` を使います。括弧の中に条件式を記述してください。

たとえば、LotArea が 15000以上 という条件にするなら、`dataset.query('LotArea >= 15000')` です。最後に `.head()` を追記すれば、今までどおり5件だけの取得になります。

参考：<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.query.html>
[\(http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.query.html\)](http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.query.html)

In [15]:



```
# 'YearBuilt' が2000以降の物件のみを抽出し、最初の5件のみ表示
dataset=pd.DataFrame(housing_data)
dataset=dataset.query('YearBuilt>=2000')
#print(dataset['YearBuilt'])
dataset.head()
```

Out[15]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	L
0	1	60	RL	65.0	8450	Pave	NaN	Reg		Lvl
2	3	60	RL	68.0	11250	Pave	NaN	IR1		Lvl
4	5	60	RL	84.0	14260	Pave	NaN	IR1		Lvl
6	7	20	RL	75.0	10084	Pave	NaN	Reg		Lvl
11	12	60	RL	85.0	11924	Pave	NaN	IR1		Lvl

5 rows × 82 columns

dataset.query('LotArea >= 15000 and MSSubClass >= 50') のように複数の条件を指定することも可能です。

In [16]:



```
# 'YearBuilt' が2000以降、'GarageCars' が2以上の物件を抽出
dataset=pd.DataFrame(housing_data)
dataset=dataset.query('YearBuilt >= 2000 and GarageCars >= 2')
#print(dataset['YearBuilt'], dataset['GarageCars'])
dataset.head()
```

Out[16]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	L
0	1	60	RL	65.0	8450	Pave	NaN	Reg		Lvl
2	3	60	RL	68.0	11250	Pave	NaN	IR1		Lvl
4	5	60	RL	84.0	14260	Pave	NaN	IR1		Lvl
6	7	20	RL	75.0	10084	Pave	NaN	Reg		Lvl
11	12	60	RL	85.0	11924	Pave	NaN	IR1		Lvl

5 rows × 82 columns

5. データの可視化

データを理解するには要約統計量や抽出したデータを確認するだけでは不十分であり、可視化が必要です。

ヒストグラム

連続変数の分布を確認する際に有効です。DataFrameの `hist()` が使えます。

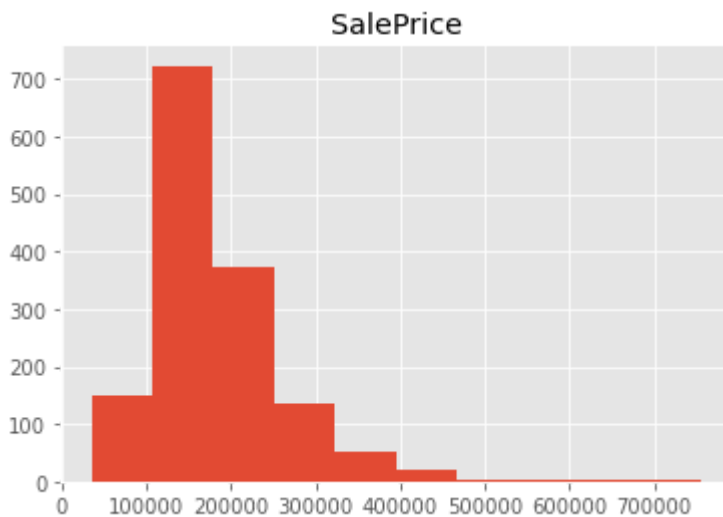
参考：<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.hist.html>
(<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.hist.html>)

In [18]:

```
# datasetの'SalePrice'をヒストグラムで表示
housing_data = pd.read_csv("kaggle_housing_price.csv")
df=pd.DataFrame(housing_data)
df.hist('SalePrice')
```

Out[18]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000022EDE306B80>]],
      dtype=object)
```



散布図

2つの変数の関係性を確認する際に有効です。DataFrameの `plot()` が使えます。

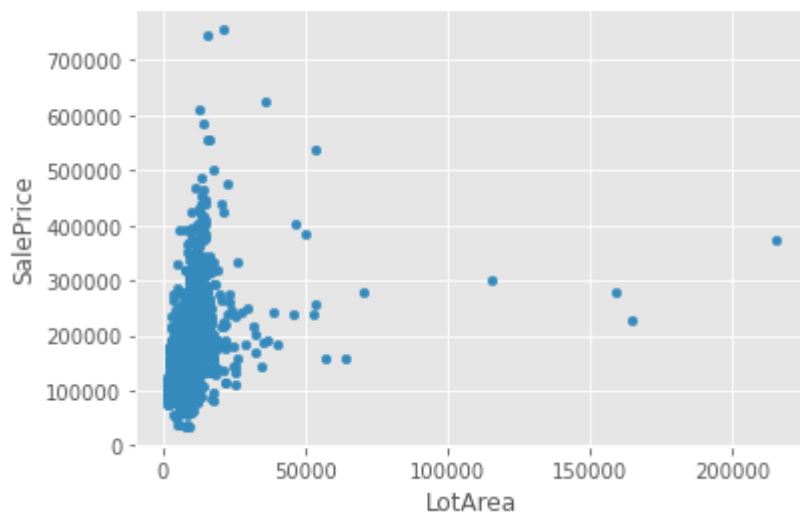
参考：<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html>
(<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html>)

In [19]:

```
# datasetの'LotArea'と'SalePrice'を散布図で表示  
df=pd.DataFrame(housing_data)  
df.plot(kind='scatter', x='LotArea', y='SalePrice')
```

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x22edea45d30>



棒グラフ

大小や増減を比較する際に有効です。DataFrameの `plot.bar()` が使えます。

参考：<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.bar.html>
(<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.bar.html>)

In [22]:

```
# 'SalePrice' のSaleCondition毎の平均を変数 price_by_conditionに格納
df=pd.DataFrame(housing_data)
grouped = df.groupby('SaleCondition')

price_by_condition=grouped.mean().SalePrice

df=pd.DataFrame(price_by_condition)

# price_by_conditionが持つ、棒グラフを表示する命令を実行
df.plot.bar()
```

Out[22]:

<matplotlib.axes._subplots.AxesSubplot at 0x22edfb8d340>



箱ヒゲ図 (Boxplot)

複数の変数の分布を比較する際に有効です。（棒グラフでは平均の比較はできますが、分布全体の比較はできません）

DataFrameの `boxplot()` が使えます。

参考：<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.boxplot.html>
(<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.boxplot.html>)

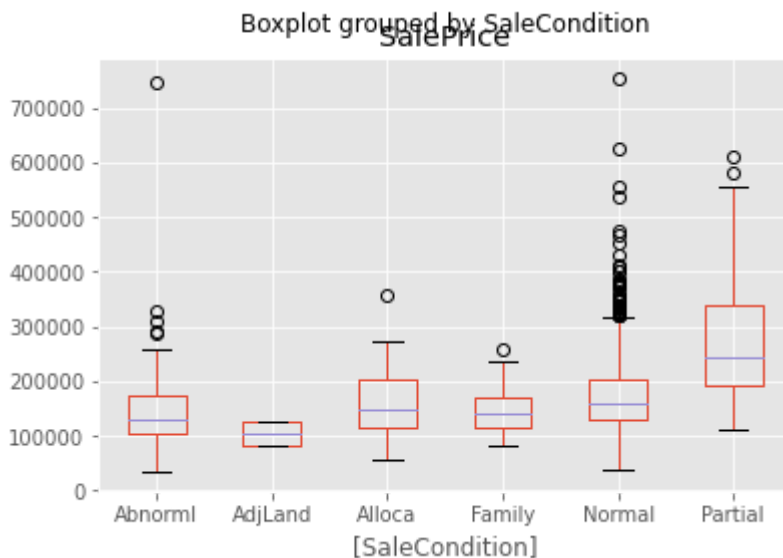
In [23]:

```
h_data=housing_data.loc[0:,:,"SaleCondition","SalePrice"]
df=pd.DataFrame(h_data)

# datasetの'SaleCondition'ごとに'SalePrice'をboxplotで表示
df.boxplot(by='SaleCondition')
```

Out[23]:

<matplotlib.axes._subplots.AxesSubplot at 0x22edfbf27f0>



6. 欠損値の確認

通常の集めたデータは、一部のデータが欠けていることがほとんどです。

欠けているデータを **欠損値** といいます。データに欠損値があると、演算でエラーが起きる場合があります。

欠損値の扱いは欠損が発生した原因により異なります。基本的には、精度に影響しないようなデータで埋めます。一例としては 0、平均値や中央値などです。

ある列が欠損値を持っているかどうかは Pandas の `isnull()` でわかります。

`pd.isnull(dataset['LotFrontage'])` のように記述します。また、`.sum()` をつなげることで、その列で欠損値を持つ行数がわかります。

参考：<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.isnull.html>
<http://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.isnull.html>

In [24]:



```
# 列ごとに欠損値の有無を確認
df=pd.DataFrame(housing_data)
print(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
```

課題1 'YearBuilt'が2005年以降、'GarageCars'が2以上の物件を抽出してください

In [17]:



```
# 'YearBuilt'が2005以降、'GarageCars'が2以上の物件を抽出
dataset=pd.DataFrame(housing_data)
dataset=dataset.query('YearBuilt >= 2005 and GarageCars >= 2')
#print(dataset['YearBuilt'], dataset['GarageCars'])
dataset.head()
```

Out[17]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
11	12	60	RL	85.0	11924	Pave	NaN	IR1	Lvl
13	14	20	RL	91.0	10652	Pave	NaN	IR1	Lvl
20	21	60	RL	101.0	14215	Pave	NaN	IR1	Lvl
25	26	20	RL	110.0	14230	Pave	NaN	Reg	Lvl
27	28	20	RL	98.0	11478	Pave	NaN	Reg	Lvl

5 rows × 82 columns

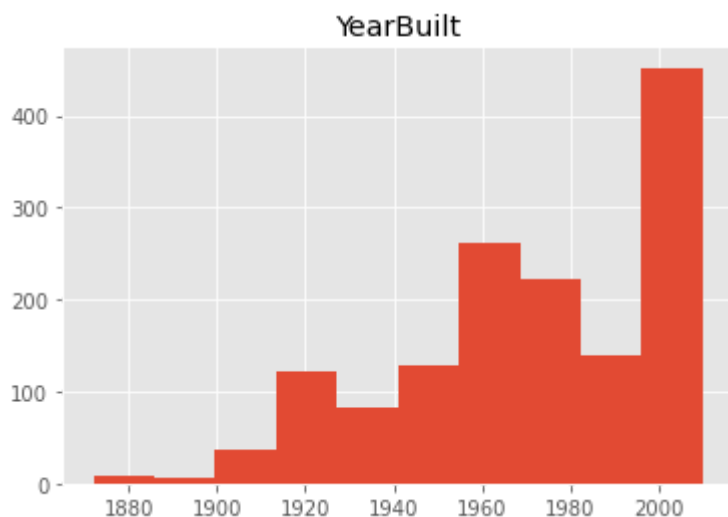
課題2 datasetの'YearBuilt'をヒストグラムで表示してください

In [20]:

```
# datasetの'YearBuilt'をヒストグラムで表示
housing_data = pd.read_csv("kaggle_housing_price.csv")
df=pd.DataFrame(housing_data)
df.hist('YearBuilt')
```

Out[20]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000022EDFAB0DC0>]],
      dtype=object)
```



課題3 datasetの'LotFrontage'と'SalePrice'の関係を散布図で表示してください

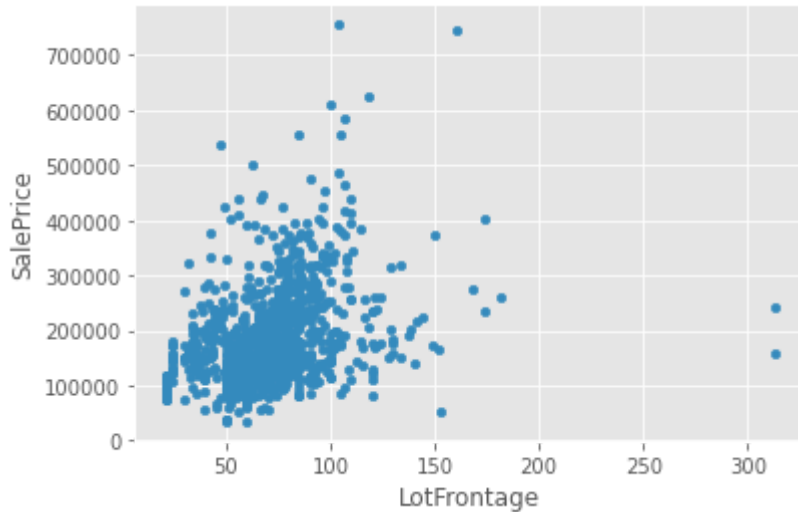
In [21]:



```
# datasetの'LotFrontage' と 'SalePrice' を散布図で表示  
df=pd.DataFrame(housing_data)  
df.plot(kind='scatter', x='LotFrontage', y='SalePrice')
```

Out[21]:

<matplotlib.axes._subplots.AxesSubplot at 0x22edfb2d820>



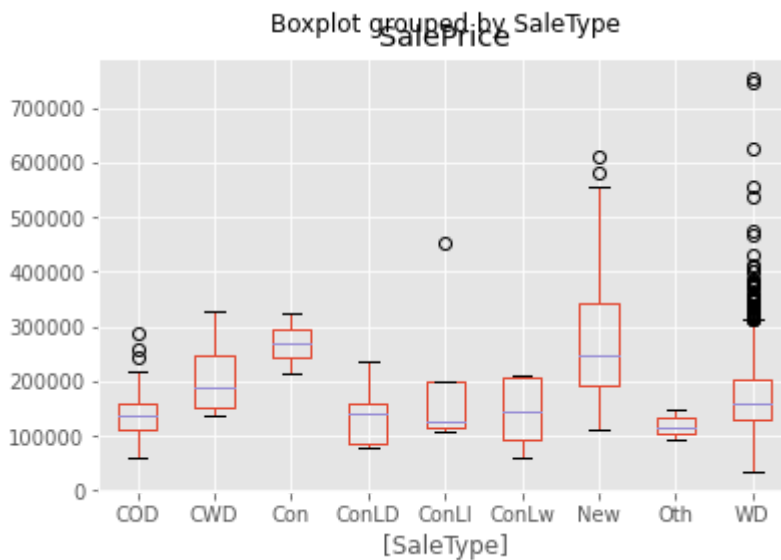
課題4 datasetの'SaleType'ごとに'SalePrice'を箱ひげ図で表示してください

In [26]:

```
h_data=housing_data.loc[0:,: ,["SaleType","SalePrice"]]  
df=pd.DataFrame(h_data)  
  
# datasetの'SaleType'ごとに'SalePrice'をboxplotで表示  
df.boxplot(by='SaleType')
```

Out[26]:

<matplotlib.axes._subplots.AxesSubplot at 0x22edfce51c0>



In []: