# final

December 15, 2018

## 1 DATA1030 Final

#### 1.0.1 Due 12/15/18 at 11:59 pm

---

**Student Name: John Facey**

**Student Email: john_facey@brown.edu**

**Student Github Name: jfacey16**

**Link to student Github Account: https://github.com/BrownDSI/f18-data1030-jfacey16**

**Student Kaggle Account Name: jfacey**

**Link to student Kaggle Account: https://www.kaggle.com/jfacey**

---

Directions: 1. This is an open computer, open book, and open web exam. You are encouraged to review concepts from lectures, labs and the textbooks for definitions and technical help. 2. However, you are expressly forbidden from searching for actual or similar problem solutions. 3. All work on this exam must be entirely your own. No talking or sharing with your classmates or anyone else. 4. You can use PyCharm, Pythontutor and any other tools you like to work on various problems in this exam. 5. Submission: Create a directory called `final` at the top level of your data1030 student GitHub folder.
6. Place a notebook called `final.ipynb` in it that contains the exam tasks below. Include all necessary code. 7. Be sure to organize your notebook so that it is clear what each cell is doing, and which question it relates to or answers. 8. **Important:** your `final` directory must include all additional files that your notebook requires. ** The grading process automatically uses the file names provided, so please spell and capitalize them exactly as given.**

**Notebooks that cannot be run from start to finish will be scored a zero.**

## 1.1 Guided Kaggle Competition

For your final exam in DATA 1030, you will create a submission for the House Prices: Advanced Regression Techniques description Kaggle competition.

In order to speed your work, and to give you an example of a detailed analysis of this dataset that leads to a reasonable model, your work will be guided by Erik Bruin's kernel analysis and submission described in this Kaggle R Kernel House prices: Lasso, XGBoost, and a detailed EDA.

Below is a linked table of contents to a copy of this kernel.

Your task for this exam will be to use Python, sklearn and the plloting libraries of your choice, to recreate the critical aspects of his analysis (including ETL and EDA) in order for you to develop your own submission. While it is possible to work online using the Kaggle platform, it will probably be more efficient for you to work locally by modifying this notebook.

## 1.2 Below are the required sections for your notebook.

- Include an appropriate narratives where appropriate, also try and fully develop most of the techniques he employed to visualize, analyze and improve the data.

- Along the way be sure to do appropriate ETL on the final model variables model, but in order to save time, you can skip data cleaning and other steps on irrelevant variables.

- For Section 9, you should use sklearn gridsearch to try and improve his final model.

- Extra Credit [10]: Review the sklearn Preprocessing Material and implement your feature transformations using appropriate Transformer functions, e.g. the preprocessing module further provides a utility class StandardScaler that implements the Transformer API to compute the mean and standard deviation on a training set so as to be able to later reapply the same transformation on the testing set.

### 1.2.1 Final Hand-in steps:

**Kaggle competition entry**

- Design your notebook so that when run top to bottom it will generate a copy of your final `final_submission.csv`. Include a copy of this file in the **`final`** directory that you turn in.
- Save a copy of the final copy of your notebook as a regular `.ipynb` and as a `.pdf`
- Remeber to also participate in the competition and to submit your final submission.

### 1.2.2 Additional Resources:

The Kaggle machine learning tutorial is quite good, and also uses the Ames Housing dataset in many of its kernels. For example,

- https://www.kaggle.com/dansbecker/xgboost
- https://www.kaggle.com/dansbecker/submitting-from-a-kernel

You are also encouraged to look at, as needed, at the other kernels related to this competition (even the ones in Python)

---

```
### BEGIN SOLUTION
```

## 1.3 1 Executive Summary [10]

## 1.4 2 Introduction

## 1.5 3 Loading and Exploring Data [10]

### 1.5.1 3.1 Loading libraries required and reading the data into Python

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import plotly
        import plotly.plotly as py
        import plotly.graph_objs as go
        #import sklearn
        from scipy import stats
        plotly.offline.init_notebook_mode(connected=True)
```

```
In [2]: train_df = pd.read_csv("all/train.csv")
        test_df = pd.read_csv("all/test.csv")
```

### 1.5.2 3.2 Data size and structure

```
In [3]: train_df.head()
```

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

          LandContour Utilities    ...     PoolArea PoolQC Fence MiscFeature MiscVal  \
        0         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
        1         Lvl    AllPub    ...            0    NaN   NaN         NaN       0
```

```
2         Lvl    AllPub    ...            0    NaN   NaN          NaN        0
3         Lvl    AllPub    ...            0    NaN   NaN          NaN        0
4         Lvl    AllPub    ...            0    NaN   NaN          NaN        0

  MoSold YrSold  SaleType  SaleCondition  SalePrice
0      2   2008        WD         Normal     208500
1      5   2007        WD         Normal     181500
2      9   2008        WD         Normal     223500
3      2   2006        WD        Abnorml     140000
4     12   2008        WD         Normal     250000

[5 rows x 81 columns]
```

```
In [4]: train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
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
OverallQual      1460 non-null int64
OverallCond      1460 non-null int64
YearBuilt        1460 non-null int64
YearRemodAdd     1460 non-null int64
RoofStyle        1460 non-null object
RoofMatl         1460 non-null object
Exterior1st      1460 non-null object
Exterior2nd      1460 non-null object
MasVnrType       1452 non-null object
MasVnrArea       1452 non-null float64
ExterQual        1460 non-null object
ExterCond        1460 non-null object
Foundation       1460 non-null object
```

```
BsmtQual         1423 non-null object
BsmtCond         1423 non-null object
BsmtExposure     1422 non-null object
BsmtFinType1     1423 non-null object
BsmtFinSF1       1460 non-null int64
BsmtFinType2     1422 non-null object
BsmtFinSF2       1460 non-null int64
BsmtUnfSF        1460 non-null int64
TotalBsmtSF      1460 non-null int64
Heating          1460 non-null object
HeatingQC        1460 non-null object
CentralAir       1460 non-null object
Electrical       1459 non-null object
1stFlrSF         1460 non-null int64
2ndFlrSF         1460 non-null int64
LowQualFinSF     1460 non-null int64
GrLivArea        1460 non-null int64
BsmtFullBath     1460 non-null int64
BsmtHalfBath     1460 non-null int64
FullBath         1460 non-null int64
HalfBath         1460 non-null int64
BedroomAbvGr     1460 non-null int64
KitchenAbvGr     1460 non-null int64
KitchenQual      1460 non-null object
TotRmsAbvGrd     1460 non-null int64
Functional       1460 non-null object
Fireplaces       1460 non-null int64
FireplaceQu      770 non-null object
GarageType       1379 non-null object
GarageYrBlt      1379 non-null float64
GarageFinish     1379 non-null object
GarageCars       1460 non-null int64
GarageArea       1460 non-null int64
GarageQual       1379 non-null object
GarageCond       1379 non-null object
PavedDrive       1460 non-null object
WoodDeckSF       1460 non-null int64
OpenPorchSF      1460 non-null int64
EnclosedPorch    1460 non-null int64
3SsnPorch        1460 non-null int64
ScreenPorch      1460 non-null int64
PoolArea         1460 non-null int64
PoolQC           7 non-null object
Fence            281 non-null object
MiscFeature      54 non-null object
MiscVal          1460 non-null int64
MoSold           1460 non-null int64
YrSold           1460 non-null int64
```

```
SaleType         1460 non-null object
SaleCondition    1460 non-null object
SalePrice        1460 non-null int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

```
In [5]: test_df.head()
```

```
Out[5]:      Id  MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
        0  1461          20       RH         80.0    11622   Pave   NaN      Reg
        1  1462          20       RL         81.0    14267   Pave   NaN      IR1
        2  1463          60       RL         74.0    13830   Pave   NaN      IR1
        3  1464          60       RL         78.0     9978   Pave   NaN      IR1
        4  1465         120       RL         43.0     5005   Pave   NaN      IR1

          LandContour Utilities      ...       ScreenPorch PoolArea PoolQC  Fence  \
        0         Lvl    AllPub      ...               120        0    NaN  MnPrv
        1         Lvl    AllPub      ...                 0        0    NaN    NaN
        2         Lvl    AllPub      ...                 0        0    NaN  MnPrv
        3         Lvl    AllPub      ...                 0        0    NaN    NaN
        4         HLS    AllPub      ...               144        0    NaN    NaN

          MiscFeature MiscVal MoSold  YrSold  SaleType  SaleCondition
        0         NaN       0      6    2010        WD         Normal
        1        Gar2   12500      6    2010        WD         Normal
        2         NaN       0      3    2010        WD         Normal
        3         NaN       0      6    2010        WD         Normal
        4         NaN       0      1    2010        WD         Normal

        [5 rows x 80 columns]
```

```
In [6]: test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 80 columns):
Id               1459 non-null int64
MSSubClass       1459 non-null int64
MSZoning         1455 non-null object
LotFrontage      1232 non-null float64
LotArea          1459 non-null int64
Street           1459 non-null object
Alley            107 non-null object
LotShape         1459 non-null object
LandContour      1459 non-null object
Utilities        1457 non-null object
LotConfig        1459 non-null object
LandSlope        1459 non-null object
```

```
Neighborhood     1459 non-null object
Condition1       1459 non-null object
Condition2       1459 non-null object
BldgType         1459 non-null object
HouseStyle       1459 non-null object
OverallQual      1459 non-null int64
OverallCond      1459 non-null int64
YearBuilt        1459 non-null int64
YearRemodAdd     1459 non-null int64
RoofStyle        1459 non-null object
RoofMatl         1459 non-null object
Exterior1st      1458 non-null object
Exterior2nd      1458 non-null object
MasVnrType       1443 non-null object
MasVnrArea       1444 non-null float64
ExterQual        1459 non-null object
ExterCond        1459 non-null object
Foundation       1459 non-null object
BsmtQual         1415 non-null object
BsmtCond         1414 non-null object
BsmtExposure     1415 non-null object
BsmtFinType1     1417 non-null object
BsmtFinSF1       1458 non-null float64
BsmtFinType2     1417 non-null object
BsmtFinSF2       1458 non-null float64
BsmtUnfSF        1458 non-null float64
TotalBsmtSF      1458 non-null float64
Heating          1459 non-null object
HeatingQC        1459 non-null object
CentralAir       1459 non-null object
Electrical       1459 non-null object
1stFlrSF         1459 non-null int64
2ndFlrSF         1459 non-null int64
LowQualFinSF     1459 non-null int64
GrLivArea        1459 non-null int64
BsmtFullBath     1457 non-null float64
BsmtHalfBath     1457 non-null float64
FullBath         1459 non-null int64
HalfBath         1459 non-null int64
BedroomAbvGr     1459 non-null int64
KitchenAbvGr     1459 non-null int64
KitchenQual      1458 non-null object
TotRmsAbvGrd     1459 non-null int64
Functional       1457 non-null object
Fireplaces       1459 non-null int64
FireplaceQu      729 non-null object
GarageType       1383 non-null object
GarageYrBlt      1381 non-null float64
```

```
GarageFinish     1381 non-null object
GarageCars       1458 non-null float64
GarageArea       1458 non-null float64
GarageQual       1381 non-null object
GarageCond       1381 non-null object
PavedDrive       1459 non-null object
WoodDeckSF       1459 non-null int64
OpenPorchSF      1459 non-null int64
EnclosedPorch    1459 non-null int64
3SsnPorch        1459 non-null int64
ScreenPorch      1459 non-null int64
PoolArea         1459 non-null int64
PoolQC           3 non-null object
Fence            290 non-null object
MiscFeature      51 non-null object
MiscVal          1459 non-null int64
MoSold           1459 non-null int64
YrSold           1459 non-null int64
SaleType         1458 non-null object
SaleCondition    1459 non-null object
dtypes: float64(11), int64(26), object(43)
memory usage: 912.0+ KB
```

```
In [7]: test_labels = test_df.Id
        train_df.drop(columns=['Id'], inplace = True)
        test_df.drop(columns=['Id'], inplace = True)
```

```
In [8]: all_df = pd.concat([train_df, test_df], sort=False, ignore_index=True)
        all_df.tail()
```

```
Out[8]:       MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
        2914         160       RM         21.0     1936   Pave   NaN      Reg
        2915         160       RM         21.0     1894   Pave   NaN      Reg
        2916          20       RL        160.0    20000   Pave   NaN      Reg
        2917          85       RL         62.0    10441   Pave   NaN      Reg
        2918          60       RL         74.0     9627   Pave   NaN      Reg

             LandContour Utilities LotConfig    ...     PoolArea PoolQC  Fence  \
        2914         Lvl    AllPub    Inside    ...            0    NaN    NaN
        2915         Lvl    AllPub    Inside    ...            0    NaN    NaN
        2916         Lvl    AllPub    Inside    ...            0    NaN    NaN
        2917         Lvl    AllPub    Inside    ...            0    NaN  MnPrv
        2918         Lvl    AllPub    Inside    ...            0    NaN    NaN

             MiscFeature MiscVal MoSold  YrSold  SaleType  SaleCondition  SalePrice
        2914         NaN       0      6    2006        WD         Normal        NaN
        2915         NaN       0      4    2006        WD        Abnorml        NaN
        2916          NaN       0       9    2006         WD         Abnorml          NaN
        2917         Shed     700       7    2006         WD          Normal          NaN
        2918          NaN       0      11    2006         WD          Normal          NaN

        [5 rows x 80 columns]
```

```
In [9]: all_df.shape
```

```
Out[9]: (2919, 80)
```

## 1.6 4 Exploring some of the most important variables [10]

### 1.6.1 4.1 The response variable; SalePrice

```
In [10]: data = [go.Histogram(
             x=train_df.SalePrice,
             xbins=dict(
                 start=0,
                 end=800000,
                 size=10000))]
         layout = go.Layout(
             title='Train Sale Prices',
             xaxis=dict(
                 title='SalePrice'
             ),
             yaxis=dict(
                 title='Count'
             )
         )

         plotly.offline.iplot(go.Figure(data=data,layout=layout))
```

```
In [11]: train_df.SalePrice.describe()
```

```
Out[11]: 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
```

### 1.6.2 4.2 The most important numeric predictors

#### 4.2.1 Correlations with SalePrice

```
In [12]: numeric_cols = all_df.select_dtypes(include=np.number).columns
         print('There are ' + str(numeric_cols.size) + ' numeric columns.')
```

```
There are 37 numeric columns.
```

```
In [13]: non_numeric_cols = all_df.select_dtypes(exclude=np.number).columns
         print('There are ' + str(non_numeric_cols.size) + ' non-numeric columns.')
```

```
There are 43 non-numeric columns.
```

```
In [14]: all_numeric_vars = all_df[numeric_cols]
         all_numeric_vars.shape
```

```
Out[14]: (2919, 37)
```

```
In [15]: cor_numeric_vars = all_numeric_vars.corr()
         cor_sorted = cor_numeric_vars.SalePrice.sort_values(ascending=False)
         cor_high_names = cor_sorted.index[cor_sorted > .5].tolist()

         top_numeric_vars = all_numeric_vars[cor_high_names]
         cor_top_numeric_vars = top_numeric_vars.corr()
```

```
In [16]: data = [go.Heatmap(z=cor_top_numeric_vars.values.tolist(),
                            x=cor_top_numeric_vars.columns,
                            y=cor_top_numeric_vars.columns)]
         layout=go.Layout(
             title='Top Correlations With SalePrice',
             yaxis=dict(
                 autorange='reversed',
                 automargin=True)
         )

         plotly.offline.iplot(go.Figure(data=data,layout=layout))
```

### 4.2.2 Overall Quality

```
In [17]: data = [go.Box(x=train_df.OverallQual,
                        y=train_df.SalePrice
         )]

         layout = go.Layout(
             title='Price by Quality',
             xaxis=dict(
                 title='Overall Quality'),
             yaxis=dict(
                 title='Sale Price')
         )
         plotly.offline.iplot(go.Figure(data=data,layout=layout))
```

#### 4.2.3 Above Grade (Ground) Living Area (square feet)

```
In [18]: slope, intercept, r_value, p_value, std_err = stats.linregress(train_df.GrLivArea,tra
         fit = slope*train_df.GrLivArea+intercept

         trace0 = go.Scatter(
             x = train_df.GrLivArea,
             y = train_df.SalePrice,
             mode = 'markers',
             name = 'data'

         )

         trace1 = go.Scatter(
             x = train_df.GrLivArea,
             y = fit,
             mode = 'lines',
             name = 'fit'

         )
         data = [trace0,trace1]
         layout = go.Layout(
             title = 'Price by Ground Living Area',
             xaxis = dict(
                 title = 'Ground Living Area'),
             yaxis = dict(
                 title = 'Sale Price')
         )
         plotly.offline.iplot(go.Figure(data = data, layout = layout))
```

## 1.7 5 Missing data, label encoding, and factorizing variables [5]

### 1.7.1 5.1 Completeness of the data

```
In [19]: na_cols = all_df.columns[all_df.isna().sum() > 0]
         all_df[na_cols].apply(lambda x: x.isna()).sum().sort_values(ascending=False)
```

```
Out[19]: PoolQC          2909
         MiscFeature     2814
         Alley           2721
         Fence           2348
         SalePrice       1459
         FireplaceQu     1420
         LotFrontage      486
         GarageYrBlt      159
         GarageFinish     159
         GarageQual       159
         GarageCond       159
         GarageType       157
         BsmtCond         82
         BsmtExposure     82
         BsmtQual         81
         BsmtFinType2     80
         BsmtFinType1     79
         MasVnrType       24
         MasVnrArea       23
         MSZoning          4
         BsmtFullBath      2
         BsmtHalfBath      2
         Functional        2
         Utilities         2
         BsmtFinSF2        1
         BsmtUnfSF         1
         BsmtFinSF1        1
         TotalBsmtSF       1
         SaleType          1
         KitchenQual       1
         Exterior2nd       1
         Exterior1st       1
         GarageCars        1
         GarageArea        1
         Electrical        1
         dtype: int64
```

```
In [20]: print('There are ' + str(len(na_cols)) + ' columns with missing values.')
```

```
There are 35 columns with missing values.
```

### 1.7.2 5.2 Imputing missing data

#### 5.2.1 Pool variables

```
In [21]: qualities = ['None', 'Po', 'Fa', 'TA', 'Gd', 'Ex']
```

```
In [22]: all_df.PoolQC = all_df.PoolQC.fillna('None')
         all_df.PoolQC = all_df.PoolQC.astype('category', ordered=True, categories=qualities).
         all_df.head()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[22]:    MSSubClass MSZoning  LotFrontage  LotArea Street Alley LotShape  \
         0          60       RL         65.0     8450   Pave   NaN      Reg
         1          20       RL         80.0     9600   Pave   NaN      Reg
```

```
2          60       RL         68.0    11250   Pave   NaN      IR1
3          70       RL         60.0     9550   Pave   NaN      IR1
4          60       RL         84.0    14260   Pave   NaN      IR1

  LandContour Utilities LotConfig    ...     PoolArea PoolQC Fence  \
0         Lvl    AllPub    Inside    ...            0      0   NaN
1         Lvl    AllPub       FR2    ...            0      0   NaN
2         Lvl    AllPub    Inside    ...            0      0   NaN
3         Lvl    AllPub    Corner    ...            0      0   NaN
4         Lvl    AllPub       FR2    ...            0      0   NaN

  MiscFeature MiscVal MoSold  YrSold  SaleType  SaleCondition  SalePrice
0         NaN       0      2    2008        WD         Normal   208500.0
1         NaN       0      5    2007        WD         Normal   181500.0
2         NaN       0      9    2008        WD         Normal   223500.0
3         NaN       0      2    2006        WD        Abnorml   140000.0
4         NaN       0     12    2008        WD         Normal   250000.0

[5 rows x 80 columns]
```

```
In [23]: all_df.loc[(all_df.PoolArea > 0) & (all_df.PoolQC == 0), ['PoolArea', 'PoolQC', 'Overa
```

```
Out[23]:       PoolArea  PoolQC  OverallQual
         2420       368       0            4
         2503       444       0            6
         2599       561       0            3
```

```
In [24]: all_df.PoolQC.at[2420] = 2
         all_df.PoolQC.at[2503] = 3
         all_df.PoolQC.at[2599] = 2
```

### 5.2.2 Miscellaneous Feature

```
In [25]: all_df.MiscFeature = all_df.MiscFeature.fillna('None')
         all_df.MiscFeature = all_df.MiscFeature.astype('category')
         all_df.MiscFeature.groupby(all_df.MiscFeature).count()
```

```
Out[25]: MiscFeature
         Gar2       5
         None    2814
         Othr       4
         Shed      95
         TenC       1
         Name: MiscFeature, dtype: int64
```

### 5.2.3 Alley

```
In [26]: all_df.Alley = all_df.Alley.fillna('None')
         all_df.Alley = all_df.Alley.astype('category')
         all_df.Alley.groupby(all_df.Alley).count()
```

```
Out[26]: Alley
         Grvl     120
         None    2721
         Pave      78
         Name: Alley, dtype: int64
```

### 5.2.4 Fence

```
In [27]: all_df.Fence = all_df.Fence.fillna('None')
         all_df.Fence = all_df.Fence.astype('category')
         all_df.Fence.groupby(all_df.Fence).count()
```

```
Out[27]: Fence
         GdPrv     118
         GdWo      112
         MnPrv     329
         MnWw       12
         None     2348
         Name: Fence, dtype: int64
```

### 5.2.5 Fireplace variables

```
In [28]: all_df.FireplaceQu = all_df.FireplaceQu.fillna('None')
         all_df.FireplaceQu = all_df.FireplaceQu.astype('category', ordered=True, categories=qu
         all_df.FireplaceQu.groupby(all_df.FireplaceQu).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[28]: FireplaceQu
         0    1420
         1      46
         2      74
         3     592
         4     744
         5      43
         Name: FireplaceQu, dtype: int64
```

```
In [29]: len(all_df.FireplaceQu)
```

```
Out[29]: 2919
```

### 5.2.6 Lot variables

```
In [30]: all_df.LotFrontage = all_df.LotFrontage.fillna(all_df.LotFrontage.median())
In [31]: lot_shape_qualities = ['IR3', 'IR2', 'IR1', 'Reg']
         all_df.LotShape = all_df.LotShape.astype('category', ordered=True, categories=lot_shap
         all_df.LotShape.groupby(all_df.LotShape).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[31]: LotShape
         0      16
         1      76
         2     968
         3    1859
         Name: LotShape, dtype: int64
```

```
In [32]: all_df.LotConfig = all_df.LotConfig.astype('category')
         all_df.LotConfig.groupby(all_df.LotConfig).count()
```

```
Out[32]: LotConfig
         Corner      511
         CulDSac     176
         FR2          85
         FR3          14
         Inside     2133
         Name: LotConfig, dtype: int64
```

### 5.2.7 Garage variables

```
In [33]: all_df.GarageYrBlt = all_df.GarageYrBlt.fillna(all_df.YearBuilt)
```

```
In [34]: all_df.loc[~(all_df.GarageType.isna()) & (all_df.GarageFinish.isna()), ['GarageCars',
```

```
Out[34]:       GarageCars  GarageArea GarageType GarageCond GarageQual GarageFinish
         2126         1.0       360.0     Detchd        NaN        NaN          NaN
         2576         NaN         NaN     Detchd        NaN        NaN          NaN
```

```
In [35]: all_df.at[2126,'GarageCond'] = all_df.GarageCond.mode().iloc[0]
         all_df.at[2126,'GarageQual'] = all_df.GarageQual.mode().iloc[0]
         all_df.at[2126,'GarageFinish'] = all_df.GarageFinish.mode().iloc[0]
```

```
In [36]: all_df.GarageCars.at[2576] = 0
         all_df.GarageArea.at[2576] = 0
         all_df.GarageType.at[2576] = np.NaN
```

```
In [37]: all_df.GarageType = all_df.GarageType.fillna('None')
         all_df.GarageType = all_df.GarageType.astype('category')
         all_df.GarageType.groupby(all_df.GarageType).count()
```

```
Out[37]: GarageType
         2Types       23
         Attchd     1723
         Basment      36
         BuiltIn     186
         CarPort      15
         Detchd      778
         None        158
         Name: GarageType, dtype: int64
```

```
In [38]: finish_cats = ['None', 'Unf', 'RFn', 'Fin']
         all_df.GarageFinish = all_df.GarageFinish.fillna('None')
         all_df.GarageFinish = all_df.GarageFinish.astype('category', ordered=True, categories=
         all_df.GarageFinish.groupby(all_df.GarageFinish).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[38]: GarageFinish
         0     158
         1    1231
         2     811
         3     719
         Name: GarageFinish, dtype: int64
```

```
In [39]: all_df.GarageQual = all_df.GarageQual.fillna('None')
         all_df.GarageQual = all_df.GarageQual.astype('category', ordered=True, categories=qual
         all_df.GarageQual.groupby(all_df.GarageQual).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[39]: GarageQual
         0     158
         1       5
         2     124
         3    2605
         4      24
         5       3
         Name: GarageQual, dtype: int64
```

```
In [40]: all_df.GarageCond = all_df.GarageCond.fillna('None')
         all_df.GarageCond = all_df.GarageCond.astype('category', ordered=True, categories=qual
         all_df.GarageCond.groupby(all_df.GarageCond).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[40]: GarageCond
         0     158
         1      14
         2      74
         3    2655
         4      15
         5       3
         Name: GarageCond, dtype: int64
```

### 5.2.8 Basement Variables

```
In [41]: all_df.loc[(~(all_df.BsmtFinType1.isna())) & ((all_df.BsmtCond.isna()) | (all_df.Bsmt
```

```
Out[41]:      BsmtQual BsmtCond BsmtExposure BsmtFinType1 BsmtFinType2
         332        Gd       TA           No          GLQ          NaN
         948        Gd       TA          NaN          Unf          Unf
         1487       Gd       TA          NaN          Unf          Unf
         2040       Gd      NaN           Mn          GLQ          Rec
         2185       TA      NaN           No          BLQ          Unf
         2217      NaN       Fa           No          Unf          Unf
         2218      NaN       TA           No          Unf          Unf
         2348       Gd       TA          NaN          Unf          Unf
         2524       TA      NaN           Av          ALQ          Unf
```

```
In [42]: all_df.loc[332,'BsmtFinType2'] = all_df.BsmtFinType2.value_counts().index[1]
         all_df.loc[[949, 1488, 2349], 'BsmtExposure'] = all_df.BsmtExposure.value_counts().in
         all_df.loc[[2041, 2186, 2525], 'BsmtCond'] = all_df.BsmtCond.value_counts().index[1]
         all_df.loc[[2218, 2219], 'BsmtQual'] = all_df.BsmtQual.value_counts().index[1]
```

```
In [43]: all_df.BsmtQual = all_df.BsmtQual.fillna('None')
         all_df.BsmtQual = all_df.BsmtQual.astype('category', ordered=True, categories=qualiti
         all_df.BsmtQual.groupby(all_df.BsmtQual).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[43]: BsmtQual
         0      80
         2      87
         3    1283
```

```
        4    1211
        5     258
        Name: BsmtQual, dtype: int64
```

```
In [44]: all_df.BsmtCond = all_df.BsmtCond.fillna('None')
         all_df.BsmtCond = all_df.BsmtCond.astype('category', ordered=True, categories=qualitie
         all_df.BsmtCond.groupby(all_df.BsmtCond).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[44]: BsmtCond
         0      82
         1       5
         2     104
         3    2603
         4     125
         Name: BsmtCond, dtype: int64
```

```
In [45]: exp = ['None', 'No', 'Mn', 'Av', 'Gd']
         all_df.BsmtExposure = all_df.BsmtExposure.fillna('None')
         all_df.BsmtExposure = all_df.BsmtExposure.astype('category', ordered=True, categories=
         all_df.BsmtExposure.groupby(all_df.BsmtExposure).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[45]: BsmtExposure
         0      82
         1    1902
         2     239
         3     420
         4     276
         Name: BsmtExposure, dtype: int64
```

```
In [46]: fin_type = ['None', 'Unf', 'LwQ', 'Rec', 'BLQ', 'ALQ', 'GLQ']
         all_df.BsmtFinType1 = all_df.BsmtFinType1.fillna('None')
         all_df.BsmtFinType1 = all_df.BsmtFinType1.astype('category', ordered=True, categories=
         all_df.BsmtFinType1.groupby(all_df.BsmtFinType1).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[46]: BsmtFinType1
         0     79
         1    851
         2    154
         3    288
         4    269
         5    429
         6    849
         Name: BsmtFinType1, dtype: int64
```

```
In [47]: all_df.BsmtFinType2 = all_df.BsmtFinType2.fillna('None')
         all_df.BsmtFinType2 = all_df.BsmtFinType2.astype('category', ordered=True, categories=
         all_df.BsmtFinType2.groupby(all_df.BsmtFinType2).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[47]: BsmtFinType2
         0      79
         1    2493
         2      87
         3     106
         4      68
         5      52
         6      34
         Name: BsmtFinType2, dtype: int64
```

```
In [48]: all_df.BsmtFullBath = all_df.BsmtFullBath.fillna(0)
         all_df.BsmtHalfBath = all_df.BsmtHalfBath.fillna(0)
         all_df.BsmtFinSF1 = all_df.BsmtFinSF1.fillna(0)
         all_df.BsmtFinSF2 = all_df.BsmtFinSF2.fillna(0)
         all_df.BsmtUnfSF = all_df.BsmtUnfSF.fillna(0)
         all_df.TotalBsmtSF = all_df.TotalBsmtSF.fillna(0)
```

### 5.2.9 Masonry variables

```
In [49]: all_df.loc[~(all_df.MasVnrArea.isna()) & (all_df.MasVnrType.isna()), ['MasVnrArea', 'M
```

```
Out[49]:       MasVnrArea MasVnrType
         2610       198.0        NaN
```

```
In [50]: all_df.at[2610,'MasVnrType'] = all_df.MasVnrType.value_counts().index[1]
```

```
In [51]: mas_types = ['BrkCmn', 'BrkFace', 'Stone']
         all_df.MasVnrType = all_df.MasVnrType.fillna('None')
         all_df.MasVnrType = all_df.MasVnrType.astype('category', ordered=True, categories=mas_
         all_df.loc[all_df.MasVnrType == -1,'MasVnrType'] = 0
         all_df.MasVnrType.groupby(all_df.MasVnrType).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[51]: MasVnrType
         0    1790
         1     880
         2     249
         Name: MasVnrType, dtype: int64
```

```
In [52]: all_df.MasVnrArea = all_df.MasVnrArea.fillna(0)
```

### 5.2.10 MS Zoning

```
In [53]: all_df.MSZoning = all_df.MSZoning.fillna(all_df.MSZoning.mode().iloc[0])
         all_df.MSZoning = all_df.MSZoning.astype('category')
         all_df.MSZoning.groupby(all_df.MSZoning).count()
```

```
Out[53]: MSZoning
         C (all)      25
         FV          139
         RH           26
         RL         2269
         RM          460
         Name: MSZoning, dtype: int64
```

### 5.2.11 Kitchen variables

```
In [54]: all_df.KitchenQual = all_df.KitchenQual.fillna(all_df.KitchenQual.mode().iloc[0])
         all_df.KitchenQual = all_df.KitchenQual.astype('category', ordered=True, categories=qu
         all_df.KitchenQual.groupby(all_df.KitchenQual).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[54]: KitchenQual
         2      70
         3    1493
         4    1151
         5     205
         Name: KitchenQual, dtype: int64
```

### 5.2.12 Utilities

```
In [55]: all_df.drop(columns=['Utilities'],inplace=True)
```

### 5.2.13 Home functionality

```
In [56]: func = ['Sal', 'Sev', 'Maj2', 'Maj1', 'Mod', 'Min2', 'Min1', 'Typ']
         all_df.Functional = all_df.Functional.fillna(all_df.Functional.mode().iloc[0])
         all_df.Functional = all_df.Functional.astype('category', ordered=True, categories=func
         all_df.Functional.groupby(all_df.Functional).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[56]: Functional
         1       2
         2       9
         3      19
         4      35
         5      70
         6      65
         7    2719
         Name: Functional, dtype: int64
```

### 5.2.14 Exterior variables

```
In [57]: all_df.Exterior1st = all_df.Exterior1st.fillna(all_df.Exterior1st.mode().iloc[0])
         all_df.Exterior1st = all_df.Exterior1st.astype('category')
         all_df.Exterior1st.groupby(all_df.Exterior1st).count()
```

```
Out[57]: Exterior1st
         AsbShng      44
         AsphShn       2
         BrkComm       6
         BrkFace      87
         CBlock        2
         CemntBd     126
         HdBoard     442
         ImStucc       1
         MetalSd     450
         Plywood     221
         Stone         2
         Stucco       43
         VinylSd    1026
         Wd Sdng     411
         WdShing      56
         Name: Exterior1st, dtype: int64
```

```
In [58]: all_df.Exterior2nd = all_df.Exterior2nd.fillna(all_df.Exterior2nd.mode().iloc[0])
         all_df.Exterior2nd = all_df.Exterior2nd.astype('category')
         all_df.Exterior2nd.groupby(all_df.Exterior2nd).count()
```

```
Out[58]: Exterior2nd
         AsbShng      38
         AsphShn       4
         Brk Cmn      22
         BrkFace      47
         CBlock        3
         CmentBd     126
         HdBoard     406
         ImStucc      15
         MetalSd     447
         Other         1
         Plywood     270
         Stone         6
         Stucco       47
         VinylSd    1015
         Wd Sdng     391
         Wd Shng      81
         Name: Exterior2nd, dtype: int64
```

```
In [59]: all_df.ExterQual = all_df.ExterQual.astype('category', ordered=True, categories=quali
         all_df.ExterQual.groupby(all_df.ExterQual).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[59]: ExterQual
         2      35
         3    1798
         4     979
         5     107
         Name: ExterQual, dtype: int64
```

```
In [60]: all_df.ExterCond = all_df.ExterCond.astype('category', ordered=True, categories=quali
         all_df.ExterCond.groupby(all_df.ExterCond).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[60]: ExterCond
         1       3
         2      67
         3    2538
         4     299
         5      12
         Name: ExterCond, dtype: int64
```

### 5.2.15 Electrical system

```
In [61]: all_df.Electrical = all_df.Electrical.fillna(all_df.Electrical.mode().iloc[0])
         all_df.Electrical = all_df.Electrical.astype('category')
         all_df.Electrical.groupby(all_df.Electrical).count()
```

```
Out[61]: Electrical
         FuseA     188
         FuseF      50
         FuseP       8
         Mix         1
         SBrkr    2672
         Name: Electrical, dtype: int64
```

### 5.2.16 Sale Type and Condition

```
In [62]: all_df.SaleType = all_df.SaleType.fillna(all_df.SaleType.mode().iloc[0])
         all_df.SaleType = all_df.SaleType.astype('category')
         all_df.SaleType.groupby(all_df.SaleType).count()
```

```
Out[62]: SaleType
         COD        87
         CWD        12
         Con         5
         ConLD      26
         ConLI       9
         ConLw       8
         New       239
         Oth         7
         WD       2526
         Name: SaleType, dtype: int64
```

```
In [63]: all_df.SaleCondition = all_df.SaleCondition.astype('category')
         all_df.SaleCondition.groupby(all_df.SaleCondition).count()
```

```
Out[63]: SaleCondition
         Abnorml     190
         AdjLand      12
         Alloca       24
         Family       46
         Normal     2402
         Partial     245
         Name: SaleCondition, dtype: int64
```

## 1.8 5.3 Label encoding/factorizing the remaining character variables [5]

```
In [64]: all_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2919 entries, 0 to 2918
Data columns (total 79 columns):
MSSubClass       2919 non-null int64
MSZoning         2919 non-null category
LotFrontage      2919 non-null float64
LotArea          2919 non-null int64
Street           2919 non-null object
Alley            2919 non-null category
LotShape         2919 non-null int8
LandContour      2919 non-null object
LotConfig        2919 non-null category
LandSlope        2919 non-null object
Neighborhood     2919 non-null object
Condition1       2919 non-null object
Condition2       2919 non-null object
BldgType         2919 non-null object
HouseStyle       2919 non-null object
OverallQual      2919 non-null int64
OverallCond      2919 non-null int64
YearBuilt        2919 non-null int64
YearRemodAdd     2919 non-null int64
RoofStyle        2919 non-null object
RoofMatl         2919 non-null object
Exterior1st      2919 non-null category
Exterior2nd      2919 non-null category
MasVnrType       2919 non-null int8
MasVnrArea       2919 non-null float64
ExterQual        2919 non-null int8
ExterCond        2919 non-null int8
Foundation       2919 non-null object
BsmtQual         2919 non-null int8
BsmtCond         2919 non-null int8
BsmtExposure     2919 non-null int8
BsmtFinType1     2919 non-null int8
BsmtFinSF1       2919 non-null float64
BsmtFinType2     2919 non-null int8
BsmtFinSF2       2919 non-null float64
BsmtUnfSF        2919 non-null float64
TotalBsmtSF      2919 non-null float64
Heating          2919 non-null object
HeatingQC        2919 non-null object
CentralAir       2919 non-null object
Electrical       2919 non-null category
1stFlrSF         2919 non-null int64
2ndFlrSF         2919 non-null int64
LowQualFinSF     2919 non-null int64
GrLivArea        2919 non-null int64
```

```
BsmtFullBath     2919 non-null float64
BsmtHalfBath     2919 non-null float64
FullBath         2919 non-null int64
HalfBath         2919 non-null int64
BedroomAbvGr     2919 non-null int64
KitchenAbvGr     2919 non-null int64
KitchenQual      2919 non-null int8
TotRmsAbvGrd     2919 non-null int64
Functional       2919 non-null int8
Fireplaces       2919 non-null int64
FireplaceQu      2919 non-null int8
GarageType       2919 non-null category
GarageYrBlt      2919 non-null float64
GarageFinish     2919 non-null int8
GarageCars       2919 non-null float64
GarageArea       2919 non-null float64
GarageQual       2919 non-null int8
GarageCond       2919 non-null int8
PavedDrive       2919 non-null object
WoodDeckSF       2919 non-null int64
OpenPorchSF      2919 non-null int64
EnclosedPorch    2919 non-null int64
3SsnPorch        2919 non-null int64
ScreenPorch      2919 non-null int64
PoolArea         2919 non-null int64
PoolQC           2919 non-null int8
Fence            2919 non-null category
MiscFeature      2919 non-null category
MiscVal          2919 non-null int64
MoSold           2919 non-null int64
YrSold           2919 non-null int64
SaleType         2919 non-null category
SaleCondition    2919 non-null category
SalePrice        1460 non-null float64
dtypes: category(11), float64(12), int64(25), int8(16), object(15)
memory usage: 1.2+ MB
```

```
In [65]: char_cols = all_df.select_dtypes(include=np.object).columns
         print('There are ' + str(len(char_cols)) + ' remaining columns with character values'
```

```
There are 15 remaining columns with character values
```

### 5.3.1 Foundation

```
In [66]: all_df.Foundation = all_df.Foundation.astype('category')
         all_df.Foundation.groupby(all_df.Foundation).count()
```

```
Out[66]: Foundation
         BrkTil     311
         CBlock    1235
         PConc     1308
         Slab        49
         Stone       11
         Wood         5
         Name: Foundation, dtype: int64
```

### 5.3.2 Heating and airco

```
In [67]: all_df.Heating = all_df.Heating.astype('category')
         all_df.Heating.groupby(all_df.Heating).count()
```

```
Out[67]: Heating
         Floor       1
         GasA     2874
         GasW       27
         Grav        9
         OthW        2
         Wall        6
         Name: Heating, dtype: int64
```

```
In [68]: all_df.HeatingQC = all_df.HeatingQC.astype('category', ordered=True, categories=quali
         all_df.HeatingQC.groupby(all_df.HeatingQC).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[68]: HeatingQC
         1       3
         2      92
         3     857
         4     474
         5    1493
         Name: HeatingQC, dtype: int64
```

```
In [69]: air = ['N', 'Y']
         all_df.CentralAir = all_df.CentralAir.astype('category', ordered=True, categories=air
         all_df.CentralAir.groupby(all_df.CentralAir).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[69]: CentralAir
         0     196
         1    2723
         Name: CentralAir, dtype: int64
```

### 5.3.3 Roof

```
In [70]: all_df.RoofMatl = all_df.RoofMatl.astype('category')
         all_df.RoofMatl.groupby(all_df.RoofMatl).count()
```

```
Out[70]: RoofMatl
         ClyTile       1
         CompShg    2876
         Membran       1
         Metal         1
         Roll          1
         Tar&Grv      23
         WdShake       9
         WdShngl       7
         Name: RoofMatl, dtype: int64
```

```
In [71]: all_df.RoofStyle = all_df.RoofStyle.astype('category')
         all_df.RoofStyle.groupby(all_df.RoofStyle).count()
```

```
Out[71]: RoofStyle
         Flat         20
         Gable      2310
         Gambrel      22
         Hip         551
         Mansard      11
         Shed          5
         Name: RoofStyle, dtype: int64
```

### 5.3.4 Land

```
In [72]: all_df.LandContour = all_df.LandContour.astype('category')
         all_df.LandContour.groupby(all_df.LandContour).count()
```

```
Out[72]: LandContour
         Bnk     117
         HLS     120
         Low      60
         Lvl    2622
         Name: LandContour, dtype: int64
```

```
In [73]: slp = ['Sev', 'Mod', 'Gtl']
         all_df.LandSlope = all_df.LandSlope.astype('category', ordered=True, categories=slp).
         all_df.LandSlope.groupby(all_df.LandSlope).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype instea
```

```
Out[73]: LandSlope
         0      16
         1     125
         2    2778
         Name: LandSlope, dtype: int64
```

### 5.3.5 Dwelling

```
In [74]: all_df.BldgType = all_df.BldgType.astype('category')
         all_df.BldgType.groupby(all_df.BldgType).count()
```

```
Out[74]: BldgType
         1Fam      2425
         2fmCon      62
         Duplex     109
         Twnhs       96
         TwnhsE     227
         Name: BldgType, dtype: int64
```

```
In [75]: all_df.HouseStyle = all_df.HouseStyle.astype('category')
         all_df.HouseStyle.groupby(all_df.HouseStyle).count()
```

```
Out[75]: HouseStyle
         1.5Fin     314
         1.5Unf      19
         1Story    1471
         2.5Fin       8
         2.5Unf      24
         2Story     872
         SFoyer      83
         SLvl       128
         Name: HouseStyle, dtype: int64
```

### 5.3.6 Neighborhood and Conditions

```
In [76]: all_df.Neighborhood = all_df.Neighborhood.astype('category')
         all_df.Neighborhood.groupby(all_df.Neighborhood).count()
```

```
Out[76]: Neighborhood
         Blmngtn     28
         Blueste     10
         BrDale      30
         BrkSide    108
         ClearCr     44
         CollgCr    267
         Crawfor    103
         Edwards    194
         Gilbert    165
         IDOTRR      93
         MeadowV     37
         Mitchel    114
         NAmes      443
         NPkVill     23
         NWAmes     131
         NoRidge     71
         NridgHt    166
         OldTown    239
         SWISU       48
         Sawyer     151
         SawyerW    125
         Somerst    182
         StoneBr     51
         Timber      72
         Veenker     24
         Name: Neighborhood, dtype: int64
```

```
In [77]: all_df.Condition1 = all_df.Condition1.astype('category')
         all_df.Condition1.groupby(all_df.Condition1).count()
```

```
Out[77]: Condition1
         Artery      92
         Feedr      164
         Norm      2511
         PosA        20
         PosN        39
         RRAe        28
         RRAn        50
         RRNe         6
         RRNn         9
         Name: Condition1, dtype: int64
```

```
In [78]: all_df.Condition2 = all_df.Condition2.astype('category')
         all_df.Condition2.groupby(all_df.Condition2).count()
```

```
Out[78]: Condition2
         Artery       5
         Feedr       13
         Norm      2889
         PosA         4
         PosN         4
         RRAe         1
         RRAn         1
```

```
RRNn         2
Name: Condition2, dtype: int64
```

### 5.3.7 Pavement of Street & Driveway

```
In [79]: street = ['Grvl', 'Pave']
         all_df.Street = all_df.Street.astype('category', ordered=True, categories=street).cat
         all_df.Street.groupby(all_df.Street).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype inste
```

```
Out[79]: Street
         0      12
         1    2907
         Name: Street, dtype: int64
```

```
In [80]: pvd = ['N', 'P', 'Y']
         all_df.PavedDrive = all_df.PavedDrive.astype('category', ordered=True, categories=pvd
         all_df.PavedDrive.groupby(all_df.PavedDrive).count()
```

```
/Users/jack/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: FutureWarning:

specifying 'categories' or 'ordered' in .astype() is deprecated; pass a CategoricalDtype inste
```

```
Out[80]: PavedDrive
         0     216
         1      62
         2    2641
         Name: PavedDrive, dtype: int64
```

### 1.8.1 5.4 Changing some numeric variables into factors

```
In [81]: all_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2919 entries, 0 to 2918
Data columns (total 79 columns):
MSSubClass       2919 non-null int64
MSZoning         2919 non-null category
LotFrontage      2919 non-null float64
LotArea          2919 non-null int64
Street           2919 non-null int8
Alley            2919 non-null category
```

```
LotShape         2919 non-null int8
LandContour      2919 non-null category
LotConfig        2919 non-null category
LandSlope        2919 non-null int8
Neighborhood     2919 non-null category
Condition1       2919 non-null category
Condition2       2919 non-null category
BldgType         2919 non-null category
HouseStyle       2919 non-null category
OverallQual      2919 non-null int64
OverallCond      2919 non-null int64
YearBuilt        2919 non-null int64
YearRemodAdd     2919 non-null int64
RoofStyle        2919 non-null category
RoofMatl         2919 non-null category
Exterior1st      2919 non-null category
Exterior2nd      2919 non-null category
MasVnrType       2919 non-null int8
MasVnrArea       2919 non-null float64
ExterQual        2919 non-null int8
ExterCond        2919 non-null int8
Foundation       2919 non-null category
BsmtQual         2919 non-null int8
BsmtCond         2919 non-null int8
BsmtExposure     2919 non-null int8
BsmtFinType1     2919 non-null int8
BsmtFinSF1       2919 non-null float64
BsmtFinType2     2919 non-null int8
BsmtFinSF2       2919 non-null float64
BsmtUnfSF        2919 non-null float64
TotalBsmtSF      2919 non-null float64
Heating          2919 non-null category
HeatingQC        2919 non-null int8
CentralAir       2919 non-null int8
Electrical       2919 non-null category
1stFlrSF         2919 non-null int64
2ndFlrSF         2919 non-null int64
LowQualFinSF     2919 non-null int64
GrLivArea        2919 non-null int64
BsmtFullBath     2919 non-null float64
BsmtHalfBath     2919 non-null float64
FullBath         2919 non-null int64
HalfBath         2919 non-null int64
BedroomAbvGr     2919 non-null int64
KitchenAbvGr     2919 non-null int64
KitchenQual      2919 non-null int8
TotRmsAbvGrd     2919 non-null int64
Functional       2919 non-null int8
```

```
Fireplaces       2919 non-null int64
FireplaceQu      2919 non-null int8
GarageType       2919 non-null category
GarageYrBlt      2919 non-null float64
GarageFinish     2919 non-null int8
GarageCars       2919 non-null float64
GarageArea       2919 non-null float64
GarageQual       2919 non-null int8
GarageCond       2919 non-null int8
PavedDrive       2919 non-null int8
WoodDeckSF       2919 non-null int64
OpenPorchSF      2919 non-null int64
EnclosedPorch    2919 non-null int64
3SsnPorch        2919 non-null int64
ScreenPorch      2919 non-null int64
PoolArea         2919 non-null int64
PoolQC           2919 non-null int8
Fence            2919 non-null category
MiscFeature      2919 non-null category
MiscVal          2919 non-null int64
MoSold           2919 non-null int64
YrSold           2919 non-null int64
SaleType         2919 non-null category
SaleCondition    2919 non-null category
SalePrice        1460 non-null float64
dtypes: category(21), float64(12), int64(25), int8(21)
memory usage: 970.4 KB
```

### 5.4.1 Year and Month Sold

```
In [82]: all_df.MoSold = all_df.MoSold.astype('category')
         all_df.MoSold.groupby(all_df.MoSold).count()
```

```
Out[82]: MoSold
         1     122
         2     133
         3     232
         4     279
         5     394
         6     503
         7     446
         8     233
         9     158
         10    173
         11    142
         12    104
         Name: MoSold, dtype: int64
```

#### 5.4.2 MSSubClass

```
In [83]: all_df.MSSubClass = all_df.MSSubClass.astype('category')
         all_df.MSSubClass.groupby(all_df.MSSubClass).count()
```

```
Out[83]: MSSubClass
         20     1079
         30      139
         40        6
         45       18
         50      287
         60      575
         70      128
         75       23
         80      118
         85       48
         90      109
         120     182
         150       1
         160     128
         180      17
         190      61
         Name: MSSubClass, dtype: int64
```

## 1.9 6 Visualization of important variables [20]

### 1.9.1 6.1 Correlations again

```
In [84]: char_cols = all_df.select_dtypes(include='category').columns
         print('There are ' + str(len(char_cols)) + ' category columns')
         num_cols = all_df.select_dtypes(include='number').columns
         print('There are ' + str(len(num_cols)) + ' numeric columns')
```

```
There are 23 category columns
There are 56 numeric columns
```

```
In [85]: all_numeric_vars = all_df[num_cols]
```

```
In [86]: cor_numeric_vars = all_numeric_vars.corr()
         cor_sorted = cor_numeric_vars.SalePrice.sort_values(ascending=False)
         cor_high_names = cor_sorted.index[cor_sorted > .5].tolist()

         top_numeric_vars = all_numeric_vars[cor_high_names]
         cor_top_numeric_vars = top_numeric_vars.corr()
```

```
In [87]: data = [go.Heatmap(z=cor_top_numeric_vars.values.tolist(),
                            x=cor_top_numeric_vars.columns,
                            y=cor_top_numeric_vars.columns)]
         layout=go.Layout(
             title='Top Correlations With SalePrice',
             yaxis=dict(
                 autorange='reversed',
                 automargin=True)
         )

         plotly.offline.iplot(go.Figure(data=data,layout=layout))
```

### 1.9.2 6.2 Finding variable importance with a quick Random Forest

```
In [88]: all_df['NeighRich'] = 0

In [89]: rich = ['StoneBr', 'NridgHt', 'NoRidge']
         poor = ['MeadowV', 'IDOTRR', 'BrDale']
         a = ['MeadowV', 'IDOTRR', 'BrDale', 'StoneBr', 'NridgHt', 'NoRidge']
         all_df.loc[all_df['Neighborhood'].isin(rich), 'NeighRich'] = 2
         all_df.loc[~(all_df['Neighborhood'].isin(a)), 'NeighRich'] = 1
         all_df.loc[all_df['Neighborhood'].isin(poor), 'NeighRich'] = 0
         all_df.drop(columns='Neighborhood',inplace=True)

In [90]: char_cols = all_df.select_dtypes(include='category').columns
         print('There are ' + str(len(char_cols)) + ' category columns')
         num_cols = all_df.select_dtypes(include='number').columns
         print('There are ' + str(len(num_cols)) + ' numeric columns')

There are 22 category columns
There are 57 numeric columns


In [91]: # in order to do this correctly, i need to one-hot encode the categorical variables f
         # also need to bin neighborhood
         all_df = pd.get_dummies(all_df, columns=char_cols)
         all_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2919 entries, 0 to 2918
Columns: 226 entries, LotFrontage to SaleCondition_Partial
dtypes: float64(12), int64(24), int8(21), uint8(169)
memory usage: 1.3 MB


In [92]: from sklearn.ensemble import RandomForestRegressor as rfr

         quick_rfr = rfr(n_estimators = 100)
         quick_rfr.fit(X=all_df.loc[0:1459, all_df.columns != 'SalePrice'],y=all_df.loc[0:1459

Out[92]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
                    oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
In [93]: x_cols = all_df.columns[all_df.columns != 'SalePrice']
```

```
In [94]: importances, cols = zip(*sorted(zip(quick_rfr.feature_importances_,x_cols), reverse=T
```

```
In [95]: data = [go.Bar(
             y=importances,
             x=cols
         )]
         layout = go.Layout(
             title='Random Forest Feature Importances',
             yaxis=dict(
                 title='% increase MSE if variable is randomly permuted',
             )
         )

         plotly.offline.iplot(go.Figure(data=data,layout=layout))
```

### 6.2.1 Above Ground Living Area, and other surface related variables (in square feet)

### 6.2.2 The most important categorical variable; Neighborhood

### 6.2.3 Overall Quality, and other Quality variables

### 6.2.4 The second most important categorical variable; MSSubClass

### 6.2.5 Garage variables

```
In [96]: all_df.loc[2592, 'GarageYrBlt'] = 2007
```

### 6.2.6 Basement variables

## 1.10 7 Feature engineering [5]

### 1.10.1 7.1 Total number of Bathrooms

```
In [97]: all_df['TotBath'] = all_df.FullBath + (all_df.HalfBath*.5) + all_df.BsmtFullBath + (a
```

### 1.10.2 7.2 Adding 'House Age', 'Remodeled (Yes/No)', and IsNew variables

```
In [98]: all_df['Remod'] = np.where((all_df.YearBuilt == all_df.YearRemodAdd), 0, 1)
         all_df['Age'] = all_df.YrSold - all_df.YearRemodAdd
         all_df['IsNew'] = np.where((all_df.YrSold == all_df.YearBuilt), 1, 0)
         all_df.IsNew.groupby(all_df.IsNew).count()
```

```
Out[98]: IsNew
         0    2803
         1     116
         Name: IsNew, dtype: int64
```

```
In [99]: all_df.YrSold = all_df.YrSold.astype('category')
         all_df.YrSold.groupby(all_df.YrSold).count()
```

```
Out[99]: YrSold
         2006    619
         2007    692
         2008    622
         2009    647
         2010    339
         Name: YrSold, dtype: int64
```

### 1.10.3 7.3 Binning Neighborhood

```
In [100]: # doing this before one hot encoding above
```

### 1.10.4 7.4 Total Square Feet

```
In [101]: all_df['TotalSqFeet'] = all_df.GrLivArea + all_df.TotalBsmtSF
```

### 1.10.5 7.5 Consolidating Porch variables

```
In [102]: all_df['TotalPorchSF'] = all_df.OpenPorchSF + all_df.EnclosedPorch + all_df['3SsnPorc
```

## 1.11 8 Preparing data for modeling [20]

### 1.11.1 8.1 Dropping highly correlated variables

```
In [103]: dropVars = ['YearRemodAdd', 'GarageYrBlt', 'GarageArea', 'GarageCond', 'TotalBsmtSF'
          all_df.drop(columns=dropVars,inplace=True)
```

### 1.11.2 8.2 Removing outliers

```
In [104]: all_df.drop([423,1298],inplace=True)
```

### 1.11.3 8.3 PreProcessing predictor variables

```
In [105]: char_cols = all_df.select_dtypes(include='category').columns
          print('There are ' + str(len(char_cols)) + ' category columns')
          num_cols = all_df.select_dtypes(include='number').columns
          print('There are ' + str(len(num_cols)) + ' numeric columns')
          char_cols
```

```
There are 1 category columns
There are 224 numeric columns
```

```
Out[105]: Index(['YrSold'], dtype='object')
```

```
In [106]: char_cols
```

```
Out[106]: Index(['YrSold'], dtype='object')
```

#### 8.3.1 Skewness and normalizing of the numeric predictors

```
In [ ]: # dont have time/really complicated to do this after having to do one hot encoding and
        #tbh either all feature engineering should come before the random forest part, or it j
        # this out of order stuff is annoying because sklearn random forest cant handle catego
```

#### 8.3.2 One hot encoding the categorical variables

```
In [107]: # need to one_hot_encode for yrsold
          all_df = pd.get_dummies(all_df, columns=['YrSold'])
```

#### 8.3.3 Removing levels with few or no observations in train or test

### 1.11.4 8.4 Dealing with skewness of response variable

```
In [108]: all_df.SalePrice = np.log(all_df.SalePrice)
```

### 1.11.5 8.5 Composing train and test sets

```
In [109]: train = all_df[~(all_df.SalePrice.isna())]
          train_x = train.loc[:,train.columns != 'SalePrice']
          test = all_df[all_df.SalePrice.isna()]
          test_x = test.loc[:,test.columns != 'SalePrice']
```

## 1.12 9 Modeling [20]

### 1.12.1 9.1 Lasso regression model

```
In [110]: from sklearn.linear_model import LassoCV
          from sklearn.metrics import mean_squared_error
          reg = LassoCV(cv=5)
          las_fit = reg.fit(train_x, all_df.SalePrice[~(all_df.SalePrice.isna())])
          np.sqrt(mean_squared_error(las_fit.predict(train_x),all_df.SalePrice[~(all_df.SalePr
```

```
Out[110]: 0.18095219224955078
```

### 1.12.2 9.2 XGBoost model

```
In [111]: from xgboost import XGBRegressor

          my_model = XGBRegressor(n_estimators=100,max_depth=3, min_child_weight=4, learning_ra
          my_model.fit(X=train_x, y=all_df.SalePrice[~(all_df.SalePrice.isna())])
          np.sqrt(mean_squared_error(my_model.predict(train_x),all_df.SalePrice[~(all_df.SaleP
```

```
Out[111]: 0.1306952121691125
```

### 1.12.3 9.3 Averaging predictions

```
In [112]: results = pd.DataFrame((np.exp(las_fit.predict(test_x)) + (2*np.exp(my_model.predict
```

```
In [113]: results = results.rename(columns={list(results)[0]: 'SalePrice'})
```

```
In [115]: results.to_csv('final_submission.csv')
```

```
### END SOLUTION
```