Real_Estate

October 17, 2018

```
In [3]: #Kaggle competition for building the best model to predict/estimate the sale prices of
        #Dataset contains 80 characteristics such as location, square_footage, utilities, ect
        #Beginning showcases exploratory work
```

```
In [ ]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

```
In [354]: sample = pd.read_csv('sample_submission.csv')
          sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1459 entries, 0 to 1458
Data columns (total 2 columns):
Id           1459 non-null int64
SalePrice    1459 non-null float64
dtypes: float64(1), int64(1)
memory usage: 22.9 KB
```

```
In [4]: train = pd.read_csv('train_kaggle.csv')
```

```
In [5]: test = pd.read_csv('test_kaggle.csv')
```

```
In [6]: data = train.append(test)
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/pandas/core/frame.py:6201: FutureWa
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=True'.

To retain the current behavior and silence the warning, pass sort=False

  sort=sort)
```

```
In [358]: 1459*2
```

```
Out[358]: 2918
```

```
In [359]: O_type = data['BldgType'].dtype
```

```
In [7]: train.groupby('EnclosedPorch')['SalePrice'].mean().head()
```

```
Out[7]: EnclosedPorch
        0     186856.88099
        19    220000.00000
        20    122000.00000
        24     82000.00000
        30    104000.00000
        Name: SalePrice, dtype: float64
```

```
In [8]: object_columns = [column for column in data.columns if data[column].dtype == data['Bld
```

```
In [9]: data_object = pd.DataFrame()
```

```
In [10]: for name in object_columns:
             data_object[name] = data[name]
```

```
In [11]: data_object.describe()
```

```
Out[11]:        Alley BldgType BsmtCond BsmtExposure BsmtFinType1 BsmtFinType2  \
         count    198     2919     2837         2837         2840         2839
         unique     2        5        4            4            6            6
         top     Grvl     1Fam       TA           No          Unf          Unf
         freq     120     2425     2606         1904          851         2493

                BsmtQual CentralAir Condition1 Condition2    ...    MiscFeature  \
         count      2838       2919       2919       2919    ...            105
         unique        4          2          9          8    ...              4
         top          TA          Y       Norm       Norm    ...           Shed
         freq       1283       2723       2511       2889    ...             95

                Neighborhood PavedDrive PoolQC RoofMatl RoofStyle SaleCondition  \
         count          2919       2919     10     2919      2919          2919
         unique           25          3      3        8         6             6
         top           NAmes          Y     Ex  CompShg     Gable        Normal
         freq            443       2641      4     2876      2310          2402

                SaleType Street Utilities
         count      2918   2919      2917
         unique        9      2         2
         top          WD   Pave    AllPub
         freq       2525   2907      2916

         [4 rows x 43 columns]
```

```
In [365]: for column in object_columns:
              data[column] = data[column].replace(data[column].unique(), range(len(data[colum
In [366]: train['GarageType'].unique()

Out[366]: array(['Attchd', 'Detchd', 'BuiltIn', 'CarPort', nan, 'Basment', '2Types'],
                dtype=object)

In [367]: range(len(train['GarageType'].unique()))

Out[367]: range(0, 7)

In [368]: data['GarageType'].unique()

Out[368]: array([0, 1, 2, 3, 4, 5, 6])

In [370]: del data['Alley']
          del data['PoolQC']

In [371]: data.describe()

Out[371]:           1stFlrSF     2ndFlrSF    3SsnPorch  BedroomAbvGr     BldgType  \
          count  2919.000000  2919.000000  2919.000000   2919.000000  2919.000000
          mean   1159.581706   336.483727     2.602261      2.860226     0.460774
          std     392.362079   428.701456    25.188169      0.822693     1.088487
          min     334.000000     0.000000     0.000000      0.000000     0.000000
          25%     876.000000     0.000000     0.000000      2.000000     0.000000
          50%    1082.000000     0.000000     0.000000      3.000000     0.000000
          75%    1387.500000   704.000000     0.000000      3.000000     0.000000
          max    5095.000000  2065.000000   508.000000      8.000000     4.000000

                    BsmtCond  BsmtExposure   BsmtFinSF1   BsmtFinSF2  BsmtFinType1  \
          count  2919.000000   2919.000000  2918.000000  2918.000000   2919.000000
          mean      0.211716      0.800274   441.423235    49.582248      1.846523
          std       0.676432      1.233050   455.610826   169.205611      1.686056
          min       0.000000      0.000000     0.000000     0.000000      0.000000
          25%       0.000000      0.000000     0.000000     0.000000      0.000000
          50%       0.000000      0.000000   368.500000     0.000000      2.000000
          75%       0.000000      2.000000   733.000000     0.000000      3.000000
          max       4.000000      4.000000  5644.000000  1526.000000      6.000000

                      ...        SaleType  ScreenPorch       Street  TotRmsAbvGrd  \
          count       ...     2919.000000  2919.000000  2919.000000   2919.000000
          mean        ...        0.251799    16.062350     0.004111      6.451524
          std         ...        0.856761    56.184365     0.063996      1.569379
          min         ...        0.000000     0.000000     0.000000      2.000000
          25%         ...        0.000000     0.000000     0.000000      5.000000
          50%         ...        0.000000     0.000000     0.000000      6.000000
          75%         ...        0.000000     0.000000     0.000000      7.000000
          max         ...        9.000000   576.000000     1.000000     15.000000

                 TotalBsmtSF    Utilities   WoodDeckSF    YearBuilt  YearRemodAdd  \
          count  2918.000000  2919.000000  2919.000000  2919.000000   2919.000000
          mean   1051.777587     0.001713    93.709832  1971.312778   1984.264474
          std     440.766258     0.055510   126.526589    30.291442     20.894344
          min       0.000000     0.000000     0.000000  1872.000000   1950.000000
          25%     793.000000     0.000000     0.000000  1953.500000   1965.000000
          50%     989.500000     0.000000     0.000000  1973.000000   1993.000000
          75%    1302.000000     0.000000   168.000000  2001.000000   2004.000000
          max    6110.000000     2.000000  1424.000000  2010.000000   2010.000000

                      YrSold
          count  2919.000000
          mean   2007.792737
          std       1.314964
          min    2006.000000
          25%    2007.000000
          50%    2008.000000
          75%    2009.000000
          max    2010.000000

          [8 rows x 79 columns]
```

```
In [372]: data['GarageYrBlt'][data['GarageYrBlt'] > 2018]
```

```
Out[372]: 1132    2207.0
          Name: GarageYrBlt, dtype: float64
```

```
In [373]: #Date entries above the current year follow a pattern
          #Make outlier equal to 2007
          data['GarageYrBlt'][data['GarageYrBlt'] > 2018] = 2007
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWi
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [374]: #Change Year variables, 0 is newest year possible, higher number = older house
          k = ['YrSold', 'GarageYrBlt', 'YearBuilt', 'YearRemodAdd']
          for column in k:
              data[column] = [(2018 - i) for i in data[column]]
```

```
In [375]: data[k].describe()
```

```
Out[375]:             YrSold  GarageYrBlt    YearBuilt  YearRemodAdd
          count  2919.000000  2760.000000  2919.000000   2919.000000
          mean     10.207263    39.959058    46.687222     33.735526
          std       1.314964    25.206206    30.291442     20.894344
          min       8.000000     8.000000     8.000000      8.000000
          25%       9.000000    16.000000    17.000000     14.000000
          50%      10.000000    39.000000    45.000000     25.000000
          75%      11.000000    58.000000    64.500000     53.000000
          max      12.000000   123.000000   146.000000     68.000000
```

```
In [376]: np.std(data['1stFlrSF'])

Out[376]: 392.2948646055113

In [12]: data1 = data

In [378]: #Normalize:
          #x-min/xmax-xmin
          #0 to 1 scale

In [379]: is_null = [column for column in data1.columns if data1[column].isnull().sum() > 0]

In [380]: is_null_less = [column for column in data1[is_null].columns if data1[column].isnull(

In [381]: data1[is_null].info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2919 entries, 0 to 1458
Data columns (total 12 columns):
BsmtFinSF1      2918 non-null float64
BsmtFinSF2      2918 non-null float64
BsmtFullBath    2917 non-null float64
BsmtHalfBath    2917 non-null float64
BsmtUnfSF       2918 non-null float64
GarageArea      2918 non-null float64
GarageCars      2918 non-null float64
GarageYrBlt     2760 non-null float64
LotFrontage     2433 non-null float64
MasVnrArea      2896 non-null float64
SalePrice       1460 non-null float64
TotalBsmtSF     2918 non-null float64
dtypes: float64(12)
memory usage: 296.5 KB


In [382]: data1['BsmtFinSF1'].mean()

Out[382]: 441.4232350925291

In [ ]: #Next we fill in missing values
        #Through mean grouping tactics

In [384]: for column in data1[is_null_less].columns:
              data1[column][data1[column].isnull()] = data1[column].mean()
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWi
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
```

```
In [386]: new_null = [column for column in data1.columns if data1[column].isnull().sum() > 0]
```

```
In [387]: data1[new_null].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2919 entries, 0 to 1458
Data columns (total 4 columns):
GarageYrBlt    2760 non-null float64
LotFrontage    2433 non-null float64
MasVnrArea     2896 non-null float64
SalePrice      1460 non-null float64
dtypes: float64(4)
memory usage: 114.0 KB
```

```
In [388]: data1['GarageYrBlt'][data1['GarageYrBlt'].isnull()] = data1['YearBuilt'][data1['Garag
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWi
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  """Entry point for launching an IPython kernel.
```

```
In [389]: null_now = [column for column in data1.columns if data1[column].isnull().sum() > 0]
```

```
In [390]: data1[null_now].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2919 entries, 0 to 1458
Data columns (total 3 columns):
LotFrontage    2433 non-null float64
MasVnrArea     2896 non-null float64
SalePrice      1460 non-null float64
dtypes: float64(3)
memory usage: 171.2 KB
```

```
In [392]: data1.ix[5]['MasVnrType']
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: Deprecati
.ix is deprecated. Please use
```

```
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  """Entry point for launching an IPython kernel.
```

```
Out[392]: 5    1
          5    1
          Name: MasVnrType, dtype: int64
```

```
In [393]: data1['MasVnrArea'][data1['MasVnrArea'].isnull()] = [data1.groupby('MasVnrType')['Ma
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWi
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  """Entry point for launching an IPython kernel.
```

```
In [394]: data1['LotFrontage'][data1['LotFrontage'].isnull()] = [data1.groupby('LotShape')['Lo
```

```
/Users/charliecarrera/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWi
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
  """Entry point for launching an IPython kernel.
```

```
In [ ]: data1
```

```
In [396]: train_kaggle = data1[data1['SalePrice'].notnull()]
```

```
In [398]: test_kaggle = data1[data1['SalePrice'].isnull()]
```

```
In [400]: test_train = train_kaggle.drop('SalePrice', axis = 1)
```

```
In [ ]: #Multivariate regression is used for the predictions
```

```
In [413]: from sklearn.linear_model import LinearRegression
          L = LinearRegression()
```

```
In [414]: L.fit(train_kaggle.drop('SalePrice', axis = 1), train_kaggle['SalePrice'])
```

```
Out[414]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

```
In [415]: from sklearn.metrics import r2_score, mean_squared_error
```

```
In [416]: r2_score(train_kaggle['SalePrice'], L.predict(train_kaggle.drop('SalePrice', axis =
Out[416]: 0.8477035183629901

In [426]: mean_squared_error(train_kaggle['SalePrice'], L.predict(train_kaggle.drop('SalePrice

Out[426]: 30991.962054331976

In [418]: predictions = L.predict(test_kaggle)

In [419]: len(predictions)

Out[419]: 1459

In [420]: result = pd.DataFrame(columns = ['Id', 'SalePrice'])
          result['Id'] = test_kaggle['Id']
          result['SalePrice'] = predictions

Out[420]:       Id      SalePrice
          0  1461  103466.042422
          1  1462  167918.528337
          2  1463  177304.286094
          3  1464  203285.743053
          4  1465  186145.314042

In [422]: result.head()

Out[422]:       Id      SalePrice
          0  1461  103466.042422
          1  1462  167918.528337
          2  1463  177304.286094
          3  1464  203285.743053
          4  1465  186145.314042

In [ ]: #Convert result data to csv

In [424]: result.to_csv('SalePrice Kaggle.csv', index = False)
```