Remove Outliers

May 24, 2019

0.0.1 Remove Ourliers

In this section, We will remove outliers using different approaches. First, we will start by usig RANSAC algorithm which is able to define the inliers and outliers.

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.linear_model import RANSACRegressor     # Robust method for regression and
        from sklearn.linear_model import LinearRegression
        from sklearn.preprocessing import StandardScaler
        from sklearn.model_selection import train_test_split
        from sklearn.model_selection import cross_val_score
        from sklearn.pipeline import Pipeline
        from sklearn.model_selection import GridSearchCV
        from sklearn.svm import SVR
        from keras.models import Sequential
        from keras.layers import Dense
        from keras.wrappers.scikit_learn import KerasRegressor
        from keras.callbacks import EarlyStopping, ModelCheckpoint
        from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

        %matplotlib inline
```

```
In [3]: data = pd.read_csv('reduced_var_data.csv', index_col = 0)
        y = data['SalePrice']
        x = data.drop(labels = 'SalePrice', axis=1)
        print(data.shape)
        data.head()
```

```
(1459, 32)
```

```
Out[3]:     LotFrontage  LotArea  OverallQual  OverallCond  YearBuilt  YearRemodAdd  \
        Id
        1          65.0   8450.0          7.0          5.0     2003.0        2003.0
        2          80.0   9600.0          6.0          8.0     1976.0        1976.0
        3          68.0  11250.0          7.0          5.0     2001.0        2002.0
        4          60.0   9550.0          7.0          5.0     1915.0        1970.0
```

```
5          84.0  14260.0          8.0          5.0     2000.0       2000.0

    ExterQual  BsmtQual  BsmtCond  BsmtExposure    ...      GarageFinish  \
Id                                                 ...
1         3.0       4.0       3.0           1.0    ...               2.0
2         2.0       4.0       3.0           4.0    ...               2.0
3         3.0       4.0       3.0           2.0    ...               2.0
4         2.0       3.0       4.0           1.0    ...               1.0
5         3.0       4.0       3.0           3.0    ...               2.0

    GarageCars  GarageArea  OpenPorchSF  MoSold  MSZoning_RL  \
Id
1          2.0       548.0         61.0     2.0          1.0
2          2.0       460.0          0.0     5.0          1.0
3          2.0       608.0         42.0     9.0          1.0
4          3.0       642.0         35.0     2.0          1.0
5          3.0       836.0         84.0    12.0          1.0

    Neighborhood_Crawfor  MasVnrArea_209.0  MasVnrArea_428.0  SalePrice
Id
1                    0.0               0.0               0.0     208500
2                    0.0               0.0               0.0     181500
3                    0.0               0.0               0.0     223500
4                    1.0               0.0               0.0     140000
5                    0.0               0.0               0.0     250000

[5 rows x 32 columns]
```

0.1 Find Outliers using RANSACRegressor

In [4]:
```
# Define outliers in the data set
ransac = RANSACRegressor(LinearRegression(),
                         max_trials=1000,
                         min_samples=1000,
                         loss='absolute_loss',
                         random_state=42)

ransac.fit(x, y)
```

Out[4]:
```
RANSACRegressor(base_estimator=LinearRegression(copy_X=True, fit_intercept=True, n_jobs
         normalize=False),
        is_data_valid=None, is_model_valid=None, loss='absolute_loss',
        max_skips=inf, max_trials=1000, min_samples=1000, random_state=42,
        residual_threshold=None, stop_n_inliers=inf, stop_probability=0.99,
        stop_score=inf)
```

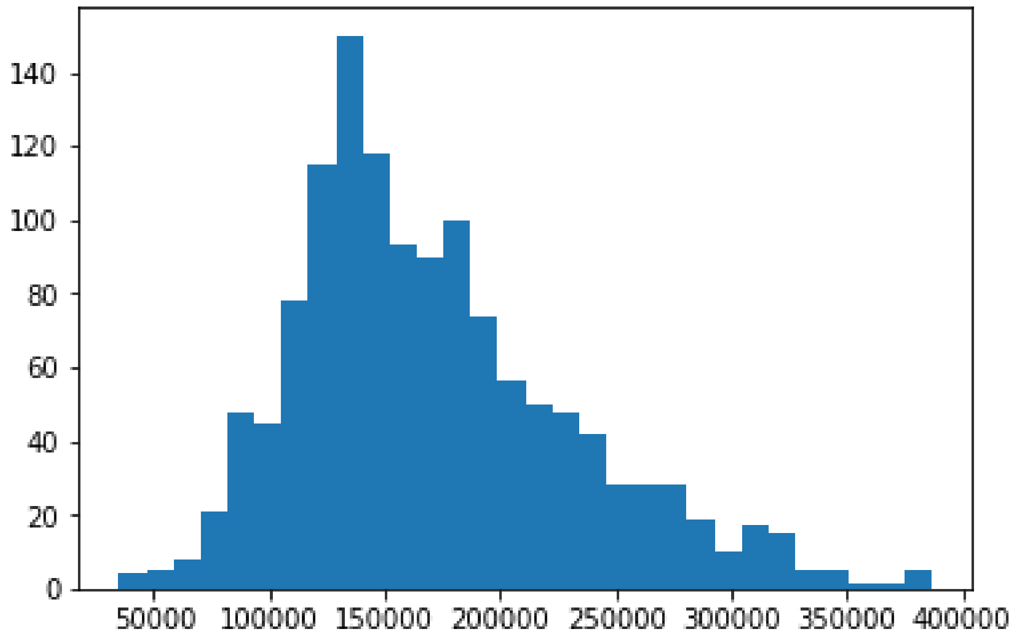
In [5]:
```
inlier_mask = ransac.inlier_mask_
outlier_mask = np.logical_not(inlier_mask)
```

```
In [6]: outlier_mask.sum(), inlier_mask.sum()
```

```
Out[6]: (152, 1307)
```

Now we removed the outliers from the dataset. The original data contains 1459 examples and the data without any outliers have 1307 examples. In other words, we removed 152 examples.

```
In [7]: plt.hist(y[inlier_mask], bins = 30);
```



0.2 SVR on data without outliers

```
In [8]: x_train, x_test, y_train, y_test = train_test_split(x[inlier_mask], y[inlier_mask], tes
```

```
In [9]: pipe = Pipeline(steps= [('ss', StandardScaler()), ('clf', SVR(gamma='scale'))])

        param_grid = {
            'clf__C':[0.1, 0.5, 1.0, 1.5, 10,100, 150, 1000],
            'clf__kernel': ['linear', 'rbf', 'sigmoid', 'poly']
        }

        search = GridSearchCV(pipe, param_grid, cv=5, iid=False, scoring='neg_mean_absolute_er
                              return_train_score=False)
        search.fit(x, y)          # Here I am using the whole training data
        print("Best parameter (CV score=%0.3f):" % search.best_score_)
        print(search.best_params_)
```

```
Best parameter (CV score=-19636.219):
{'clf__C': 1000, 'clf__kernel': 'linear'}
```

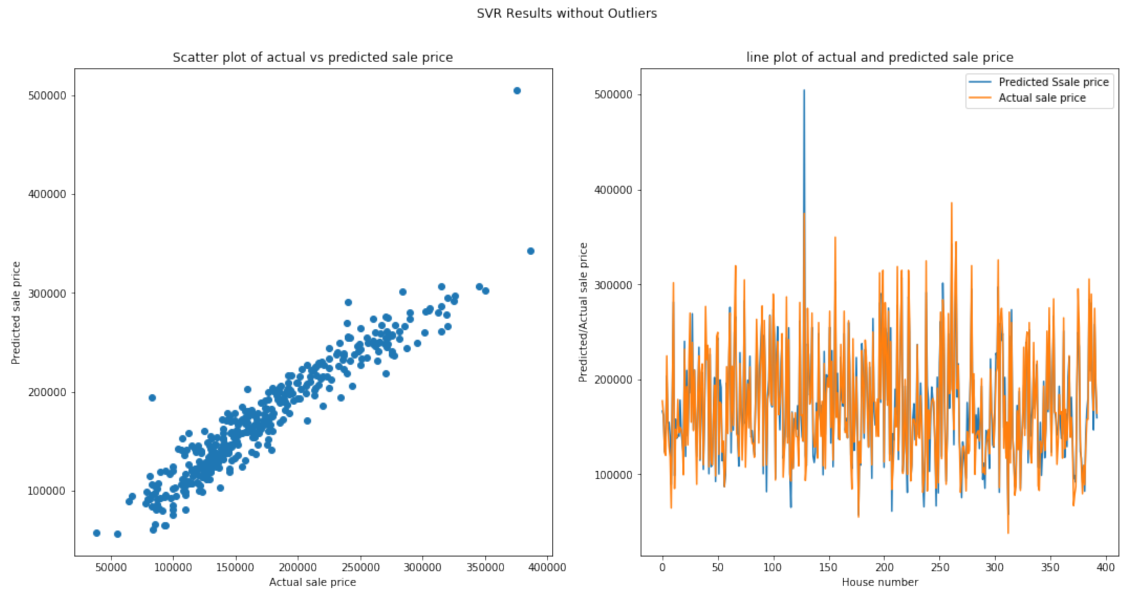
```
In [10]: ss = StandardScaler()
         ss.fit(x_train)
         x_train = ss.transform(x_train)
         x_test = ss.transform(x_test)

         best_svr = SVR(kernel='linear', gamma ='scale', C = 1000)
         best_svr.fit(x_train, y_train)
         y_pred = best_svr.predict(x_test)
         print(mean_absolute_error(y_test, y_pred))
         print('R2_score = {}'.format(r2_score(y_test, y_pred)))
```

```
14062.252899828172
R2_score = 0.9027249959650371
```

```
In [11]: print(best_svr.score(x_test, y_test))
```

```
0.9027249959650371
```

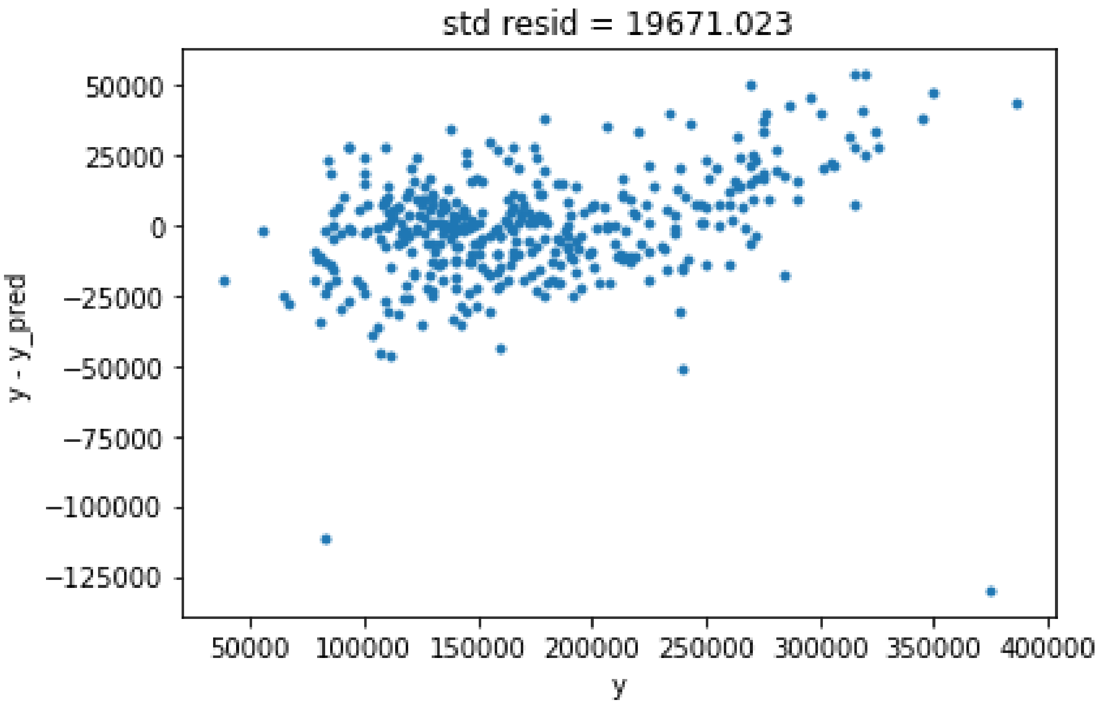
```
In [12]: y_pred = best_svr.predict(x_test)
         fig = plt.figure(figsize=(15,8))
         fig.suptitle('SVR Results without Outliers')
         plt.subplot(121)
         plt.scatter(y_test.values, y_pred)
         plt.xlabel('Actual sale price')
         plt.ylabel('Predicted sale price')
         plt.title('Scatter plot of actual vs predicted sale price')
         plt.subplot(122)
         plt.plot((y_pred), label='Predicted Ssale price')
         plt.plot((y_test.values), label='Actual sale price')
         plt.xlabel('House number')
         plt.ylabel('Predicted/Actual sale price')
         plt.title('line plot of actual and predicted sale price')
         plt.legend()
         plt.tight_layout()
         fig.subplots_adjust(top=0.88)
```




```
In [13]: print("Corrolation between true and predicted value using SVR on the actual sale pric
                format(np.corrcoef(y_test,y_pred)[0][1]))
```

```
Corrolation between true and predicted value using SVR on the actual sale price is 0.950250546
```

```
In [14]: resid = y_test - y_pred
         mean_resid = resid.mean()
         std_resid = resid.std()
         plt.plot(y_test,y_test-y_pred,'.')
         plt.xlabel('y')
         plt.ylabel('y - y_pred');
         plt.title('std resid = {:.3f}'.format(std_resid));
```



0.3 Neural Networks

```
In [15]: x_train, x_test, y_train, y_test = train_test_split(x[inlier_mask], y[inlier_mask], te

         ss = StandardScaler()
         ss.fit(x_train)
         x_train = ss.transform(x_train)
         x_test = ss.transform(x_test)
```

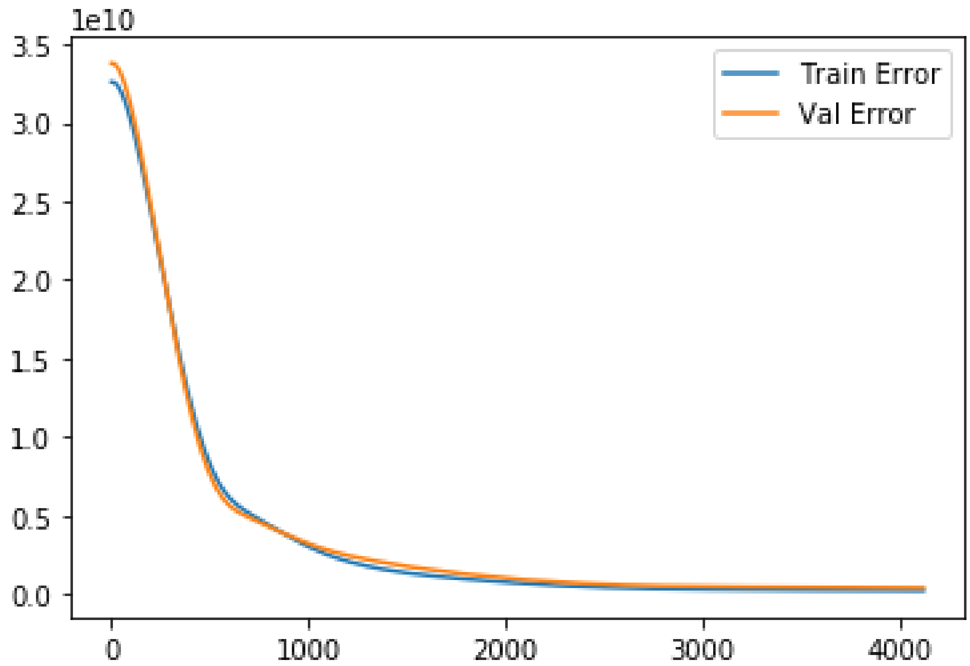
```
In [16]: seed = 42
         np.random.seed(seed)

         n_feat = x.shape[1]
         mdl = Sequential()
         mdl.add(Dense(units=256, input_dim = n_feat, activation = 'relu'))
         mdl.add(Dense(units=1, activation='linear'))

         mdl.compile(loss='mean_squared_error', optimizer='adam', metrics= ['mse', 'mae'])
         monitor = EarlyStopping(monitor= 'val_loss', min_delta=1e-3,
                                 patience = 10, verbose=1, mode = 'auto')
         history = mdl.fit(x_train, y_train, validation_data = (x_test, y_test),
                           callbacks=[monitor],batch_size= 64,epochs=5000,verbose=0)

         y_pred = mdl.predict(x_test)
```

```
MSEscore = (mean_squared_error(y_pred, y_test))
print('Score MSE = {}'.format(MSEscore))

MAEscore = (mean_absolute_error(y_pred, y_test))
print('Score MAE = {}'.format(MAEscore))
print('R2_score = {}'.format(r2_score(y_test, y_pred)))
mdl.summary()
```

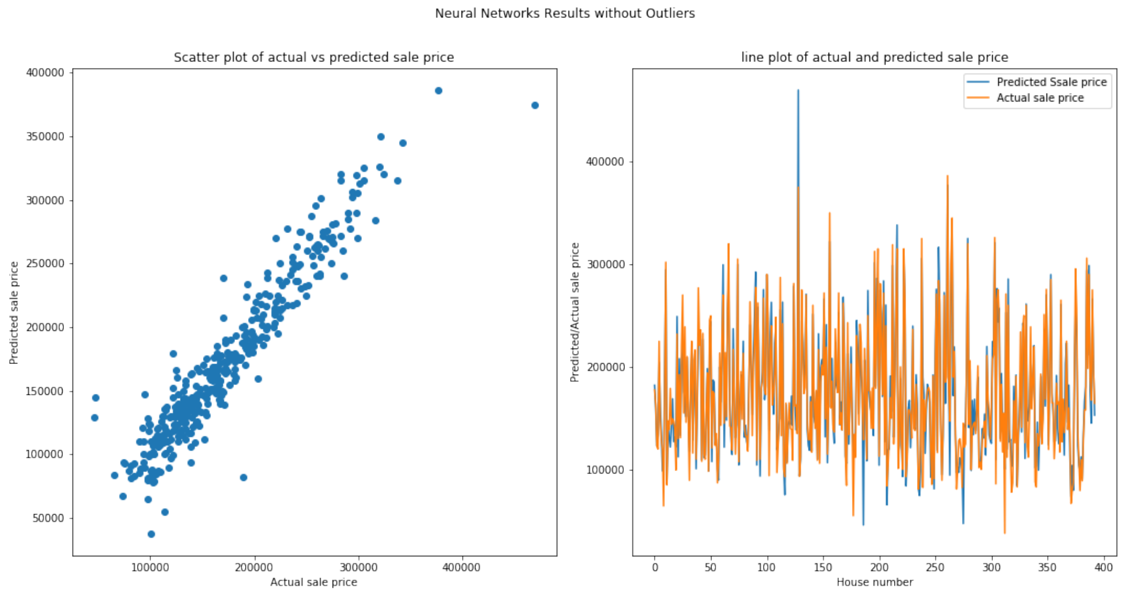
```
Epoch 04124: early stopping
Score MSE = 380631885.78712326
Score MAE = 13701.82104802799
R2_score = 0.90407417909799
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 256)               8192
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 8,449
Trainable params: 8,449
Non-trainable params: 0
_________________________________________________________________
```

In [17]:
```
plt.figure()
plt.plot(history.epoch, history.history['mean_squared_error'], label = 'Train Error')
plt.plot(history.epoch, history.history['val_mean_squared_error'], label = 'Val Error
plt.legend()
```

Out[17]:
```
<matplotlib.legend.Legend at 0x2256aa6ab00>
```



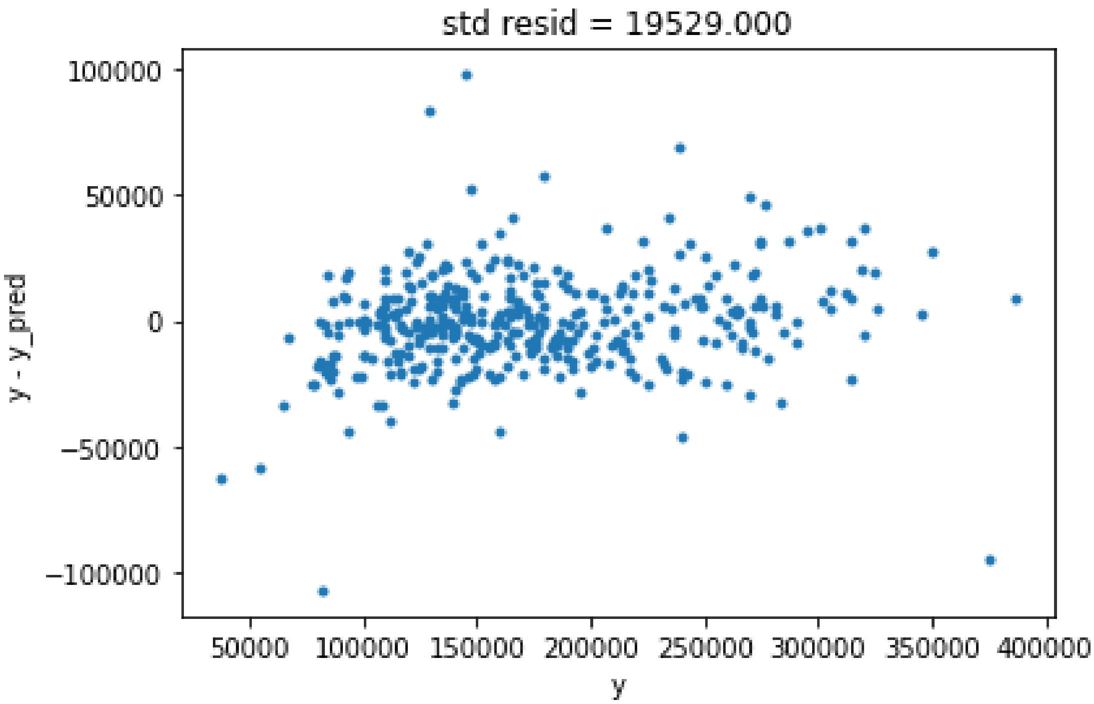

```
In [18]: y_pred = mdl.predict(x_test)
         fig = plt.figure(figsize=(15,8))
         fig.suptitle('Neural Networks Results without Outliers')
         plt.subplot(121)
         plt.scatter((y_pred), (y_test))
         plt.xlabel('Actual sale price')
         plt.ylabel('Predicted sale price')
         plt.title('Scatter plot of actual vs predicted sale price')
         plt.subplot(122)
         plt.subplots_adjust(bottom=0.25)
         plt.plot((y_pred), label='Predicted Ssale price')
         plt.plot((y_test.values), label='Actual sale price')
         plt.xlabel('House number')
         plt.ylabel('Predicted/Actual sale price')
         plt.title('line plot of actual and predicted sale price')
         plt.legend()
         plt.tight_layout()
         fig.subplots_adjust(top=0.88)
```




```
In [19]: print("Corrolation between true and predicted value using NN on the actual sale price
                 format(np.corrcoef(y_test,y_pred.squeeze())[0][1]))
```

```
Corrolation between true and predicted value using NN on the actual sale price is 0.95160458354
```

```
In [20]: resid = y_test - y_pred.squeeze()
         mean_resid = resid.mean()
         std_resid = resid.std()
         plt.plot(y_test,resid,'.')
         plt.xlabel('y')
         plt.ylabel('y - y_pred');
         plt.title('std resid = {:.3f}'.format(std_resid));
```


std resid = 19529.000
100000
50000
0
−50000
−100000
y - y_pred
50000
100000
150000
200000
250000
300000
350000
400000
y