1 Executive Summary

Outlier detection, refers to the identification of rare items, events or observations which differ from the general distribution of a population. Industry applications include fraud detection in finance, fault diagnosis in mechanics, intrusion detection in network security and pathology detection in medical imaging.

We have different outlier detection models such as **Linear Models for Outlier Detection, Proximity-Based Outlier Detection Models,Probabilistic Models for Outlier Detection,**and **outlier Ensembles and Combination frameworks**

In this notebook, I aim to compare the exisiting outlier detection models in PyOD library. PyOD contains more than 20 algorithms such as local outlier factor and neural network architectures such as autoencoders or adversarial models. Also PyOD implements combination methods for merging the results of multiple detectors and outlier ensembles which are an emerging set of models.

2 Introduction

In this notebook, first, I plan to use "Sweetviz" library in order to get some insight in to the "AmesHousing dataset". This library produces an html report for all numerical and categorical attributes, it shows missing values and their percentage and also unique values within each category. Second I use such information for EDA of the dataset. Third, I aim to show the importance of outlier detection algorithm in dealing with overfitting of a model. For this purpose, I will use and compare 10 algorithms of the PyOD library.

3 Loading and Exploring Data

3.1 Loading libraries

In [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from pandas import Series, DataFrame
from math import sqrt
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import (train_test_split, cross_val_score, RepeatedKFold)
from sklearn.ensemble import GradientBoostingRegressor
from sklearn import metrics
from scipy import stats
import matplotlib.font_manager

# Import sweetviz library
import sweetviz as sv
```

3.2 Loading Data

In [4]:
```
data = pd.read_csv('Downloads/AmesHousing.csv')
```

3.3 Data size and structure

In [5]:
```
data.shape
```

Out[5]: (2930, 82)

Our dataframe consists of 10 predictors and our response variable is "AverageTemperature".

In [6]:
```
# let us look at some records of our data
data.head()
```

Out[6]:

	Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	...	Pool Area	Pool QC	Fence	F
0	1	526301100	20	RL	141.0	31770	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	
1	2	526350040	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	...	0	NaN	MnPrv	
2	3	526351010	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	...	0	NaN	NaN	
3	4	526353030	20	RL	93.0	11160	Pave	NaN	Reg	Lvl	...	0	NaN	NaN	
4	5	527105010	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	...	0	NaN	MnPrv	

5 rows × 82 columns

4 EDA

In [7]:
```
data.isna().sum().sum()
```

Out[7]: 13997

We have had 13997 missing values in our dataset, we need to identify the features and their cause.

In [ ]:
```
# analyze the data using analyze() function
report=sv.analyze(data)

# To display the analysis of this dataframe you need to use show_html() from sweetviz libra
report.show_html()
```

The generated EDA report shows the following features have the highest percentage of missing values in our dataset.
"Alley", "Fireplace Qu","Pool QC","Fence","Misc Feature"

5 Handling Missing Data

Before handling the missing values we need to identify the number of unique values for each categorical feature. I create a dataframe which consists of the uniques values for each feature and the feature value which has the highest frequency of occurance in this dataset.

In [16]:
```
data.describe(include=[np.object]).transpose() \
    .drop("count", axis=1)
```

Out[16]:

	unique	top	freq
MS Zoning	7	RL	2273
Street	2	Pave	2918
Alley	3	No Alley	2732
Lot Shape	4	Reg	1859
Land Contour	4	Lvl	2633
Utilities	3	AllPub	2927
Lot Config	5	Inside	2140
Land Slope	3	Gtl	2789
Neighborhood	28	NAmes	443
Condition 1	9	Norm	2522
Condition 2	8	Norm	2900
Bldg Type	5	1Fam	2425
House Style	8	1Story	1481
Roof Style	6	Gable	2321
Roof Matl	8	CompShg	2887
Exterior 1st	16	VinylSd	1026
Exterior 2nd	17	VinylSd	1015
Mas Vnr Type	5	None	1752
Exter Qual	4	TA	1799
Exter Cond	5	TA	2549
Foundation	6	PConc	1310
Bsmt Qual	5	TA	1283
Bsmt Cond	5	TA	2616
Bsmt Exposure	4	No	1906
BsmtFin Type 1	6	GLQ	859
BsmtFin Type 2	6	Unf	2499
Heating	6	GasA	2885
Heating QC	5	Ex	1495
Central Air	2	Y	2734
Electrical	5	SBrkr	2682
Kitchen Qual	5	TA	1494
Functional	8	Typ	2728
Fireplace Qu	6	No Fireplace	1422
Garage Type	7	Attchd	1731
Garage Finish	4	Unf	1231
Garage Qual	6	TA	2615
Garage Cond	6	TA	2665
Paved Drive	3	Y	2652
Pool QC	5	No Pool	2917
Fence	5	No Fence	2358
Misc Feature	6	No feature	2824
Sale Type	10	WD	2536
Sale Condition	6	Normal	2413

In [17]:
```
num_missing = data.isna().sum()
num_missing = num_missing[num_missing > 0]
percent_missing = num_missing * 100 / data.shape[0]
pd.concat([num_missing, percent_missing], axis=1,
          keys=['Missing Values', 'Percentage']).\
          sort_values(by="Missing Values", ascending=False)
```

Out[17]:

	Missing Values	Percentage
Bsmt Exposure	83	2.832765
BsmtFin Type 2	81	2.764505
Bsmt Qual	80	2.730375
Bsmt Cond	80	2.730375
BsmtFin Type 1	80	2.730375
Mas Vnr Type	23	0.784983
Mas Vnr Area	23	0.784983
Bsmt Full Bath	2	0.068259
Bsmt Half Bath	2	0.068259
BsmtFin SF 1	1	0.034130
BsmtFin SF 2	1	0.034130
Bsmt Unf SF	1	0.034130
Total Bsmt SF	1	0.034130
Electrical	1	0.034130

By looking at the generated report we can see that some properties dont have the features (such as fence or fireplace) that was why the values were missing. So I substitute the missing values with a new option for that category.

In [18]:
```
data['Alley'].fillna('No Alley', inplace=True)
data["Pool QC"].fillna("No Pool", inplace=True)
data['Fence'].fillna('No Fence', inplace=True)
data['Fireplace Qu'].fillna('No Fireplace', inplace=True)
data['Misc Feature'].fillna('No feature', inplace=True)
```

"Lot Frontage" is a numerical feature and it contains 17% of our missing values. I will substitute these values with 0.

In [19]:
```
data['Lot Frontage'].fillna(0, inplace=True)
```

The report shows that we have 7 features which are related to the Garage falility. let us have a look at them.

In [20]:
```
data['Garage Cars'].fillna(0, inplace=True)
data['Garage Area'].fillna(0, inplace=True)
```

These two features are numerical and if there is no garage then we can replace them with zero

In [21]:
```
for col in ['Garage Type', 'Garage Finish', 'Garage Qual', 'Garage Cond']:
    data[col].fillna('No Garage', inplace=True)
```

If there is no Garage, then there is no quality and cond etc.

In [22]:
```
data['Garage Yr Blt'].fillna(0, inplace=True)
```

Similarly, if there is no basement or bath in a floor I subsitute them with "No Basement" and 0.

In [23]:
```
for col in ["Bsmt Exposure", "BsmtFin Type 2",
            "BsmtFin Type 1", "Bsmt Qual", "Bsmt Cond"]:
    data[col].fillna("No Basement", inplace=True)
```

In [24]:
```
for col in ["Bsmt Half Bath", "Bsmt Full Bath", "Total Bsmt SF",
            "Bsmt Unf SF", "BsmtFin SF 2", "BsmtFin SF 1"]:
    data[col].fillna(0, inplace=True)
```

In [25]:
```
data['Mas Vnr Area'].fillna(0, inplace=True)
data['Mas Vnr Type'].fillna("None", inplace=True)
data['Electrical'].fillna(data['Electrical'].mode()[0], inplace=True)
```

In [26]:
```
data.isnull().sum().sum()
```

Out[26]: 0

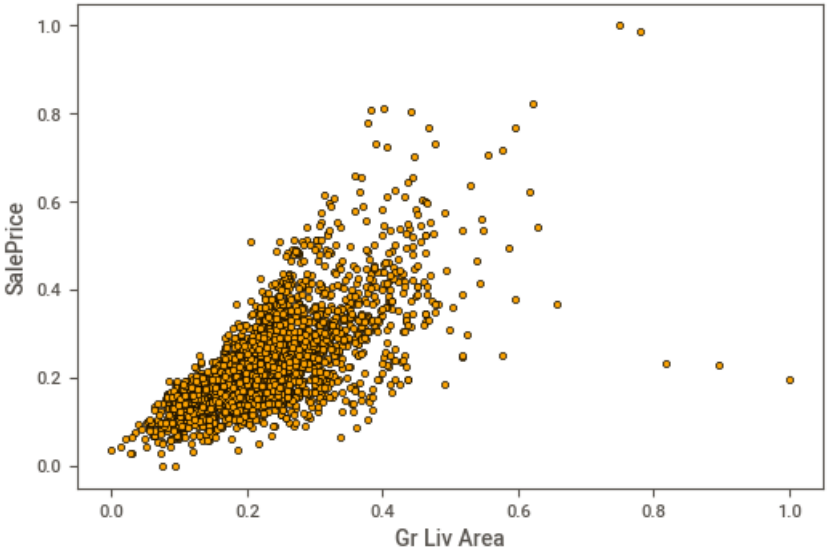
5.1 Outlier Removal

In [27]:
```
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
num_df = data.select_dtypes(include=numerics)
```

In [28]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
# We need to fit the scaler to our data before transformation
num_df = pd.DataFrame(scaler.fit_transform(num_df), columns=num_df.columns)
selected_df=num_df[['Gr Liv Area','SalePrice']]
```

In [29]:
```
plt.scatter(x=num_df['Gr Liv Area'], y=num_df['SalePrice'],
            color="orange", edgecolors="#000000", linewidths=0.5);
plt.xlabel("Gr Liv Area"); plt.ylabel("SalePrice")
```

Out[29]: Text(0, 0.5, 'SalePrice')



In [58]:
```
#Linear Models for Outlier Detection:
from pyod.models.pca import PCA
from pyod.models.mcd import MCD
from pyod.models.ocsvm import OCSVM

#Proximity-Based Outlier Detection Models:

from pyod.models.cblof import CBLOF
from pyod.models.knn import KNN
from pyod.models.lof import LOF
from pyod.models.hbos import HBOS

#Probabilistic Models for Outlier Detection:
from pyod.models.abod import ABOD

#Outlier Ensembles and Combination Frameworks
from pyod.models.feature_bagging import FeatureBagging
from pyod.models.iforest import IForest
```

In [139]:
```
random_state = np.random.RandomState(101)
outliers_fraction = 0.01

classifiers = {
        'Angle-based Outlier Detector (ABOD)': ABOD(contamination=outliers_fraction),
        'Cluster-based Local Outlier Factor (CBLOF)':CBLOF(contamination=outliers_fraction,
        'Feature Bagging':FeatureBagging(LOF(n_neighbors=35),contamination=outliers_fractio
        'Histogram-base Outlier Detection (HBOS)': HBOS(contamination=outliers_fraction),
        'Isolation Forest': IForest(contamination=outliers_fraction,random_state=random_sta
        'K Nearest Neighbors (KNN)': KNN(contamination=outliers_fraction),
        'Average KNN': KNN(method='mean',contamination=outliers_fraction),
        'PCA Outlier Detector': PCA(contamination=outliers_fraction,random_state=random_sta
        'Minimum Covariance Determinant (MCD)': MCD(contamination=outliers_fraction,random_
        'One-class SVM detector (OCSVM)': OCSVM(contamination=outliers_fraction)}
```

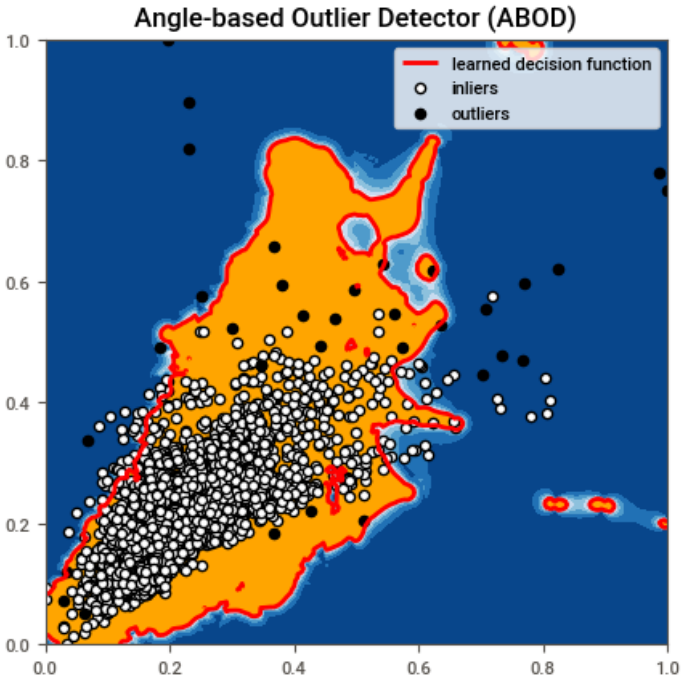
In [32]:
```
X1 = num_df['Gr Liv Area'].values.reshape(-1,1)
X2 = num_df['SalePrice'].values.reshape(-1,1)
X = np.concatenate((X1,X2),axis=1)
```

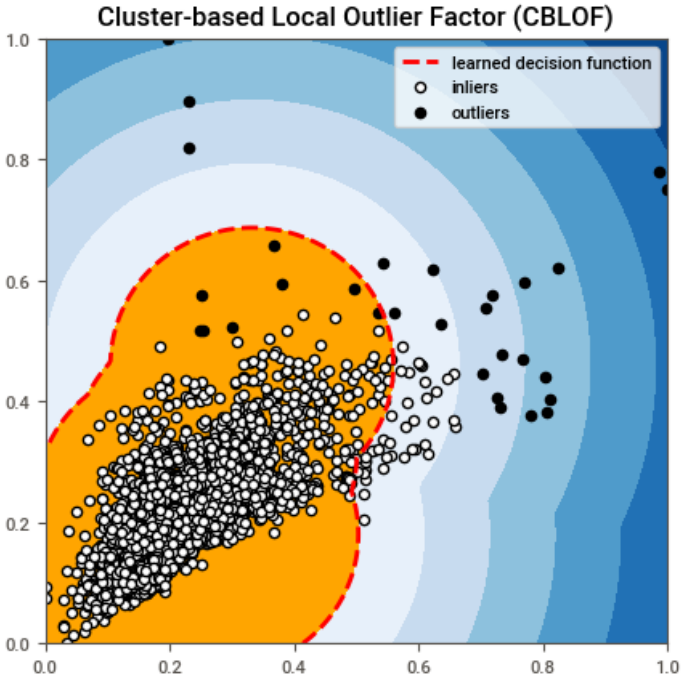
In [33]:
```
num_df.to_csv('data.csv')
```

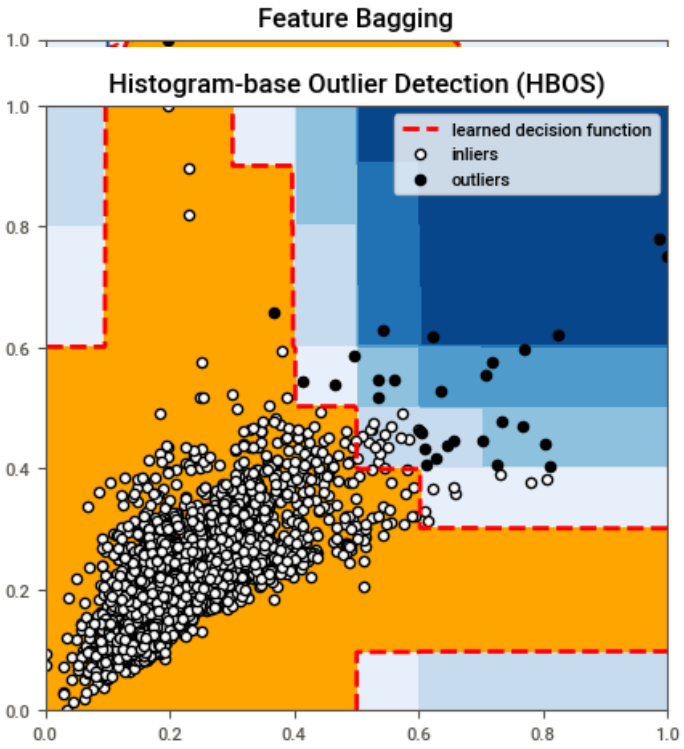
In [140]:

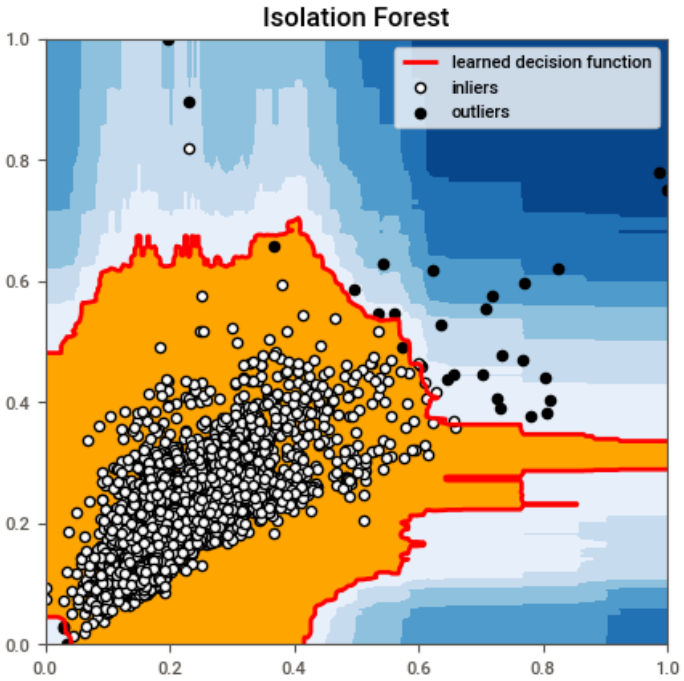
```
xx , yy = np.meshgrid(np.linspace(0,1 , 200), np.linspace(0, 1, 200))
dfxAll=num_df[['Gr Liv Area','SalePrice']]
for i, (clf_name, clf) in enumerate(classifiers.items()):
    clf.fit(X)
    scores_pred = clf.decision_function(X) * -1
    y_pred = clf.predict(X)
    n_inliers = len(y_pred) - np.count_nonzero(y_pred)
    n_outliers = np.count_nonzero(y_pred == 1)
    plt.figure(figsize=(5, 5))

    dfx = selected_df
    dfx['outlier'] = y_pred.tolist()
    dfxAll[clf_name]= y_pred.tolist()

    IX1 =  np.array(dfx['SalePrice'][dfx['outlier'] == 0]).reshape(-1,1)
    IX2 =  np.array(dfx['Gr Liv Area'][dfx['outlier'] == 0]).reshape(-1,1)

    OX1 =  dfx['SalePrice'][dfx['outlier'] == 1].values.reshape(-1,1)
    OX2 =  dfx['Gr Liv Area'][dfx['outlier'] == 1].values.reshape(-1,1)

    print('OUTLIERS : ',n_outliers,'INLIERS : ',n_inliers, clf_name)

    threshold = stats.scoreatpercentile(scores_pred,100 * outliers_fraction)

    Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) * -1
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold, 7),cmap=plt.cm.Blues_r)

    a = plt.contour(xx, yy, Z, levels=[threshold],linewidths=2, colors='red')

    plt.contourf(xx, yy, Z, levels=[threshold, Z.max()],colors='orange')

    b = plt.scatter(IX1,IX2, c='white',s=20, edgecolor='k')

    c = plt.scatter(OX1,OX2, c='black',s=20, edgecolor='k')

    plt.axis('tight')

    plt.legend(
        [a.collections[0], b,c],
        ['learned decision function', 'inliers','outliers'],
        prop=matplotlib.font_manager.FontProperties(size=8),
        loc="best")

    plt.xlim((0, 1))
    plt.ylim((0, 1))
    plt.title(clf_name)
plt.show()
```

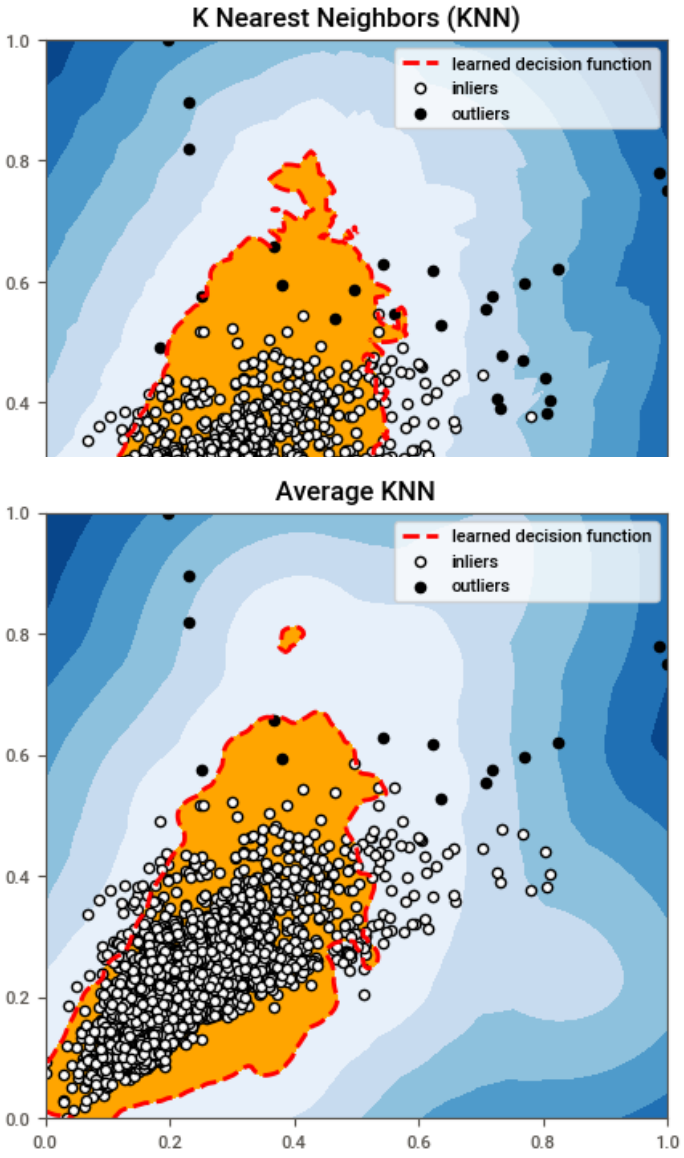
```
OUTLIERS :  35 INLIERS :  2895 Angle-based Outlier Detector (ABOD)
OUTLIERS :  30 INLIERS :  2900 Cluster-based Local Outlier Factor (CBLOF)
OUTLIERS :  30 INLIERS :  2900 Feature Bagging
OUTLIERS :  29 INLIERS :  2901 Histogram-base Outlier Detection (HBOS)
OUTLIERS :  30 INLIERS :  2900 Isolation Forest
OUTLIERS :  26 INLIERS :  2904 K Nearest Neighbors (KNN)
OUTLIERS :  15 INLIERS :  2915 Average KNN
OUTLIERS :  30 INLIERS :  2900 PCA Outlier Detector
OUTLIERS :  30 INLIERS :  2900 Minimum Covariance Determinant (MCD)
OUTLIERS :  30 INLIERS :  2900 One-class SVM detector (OCSVM)
```


Angle-based Outlier Detector (ABOD)
learned decision function
inliers
outliers
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

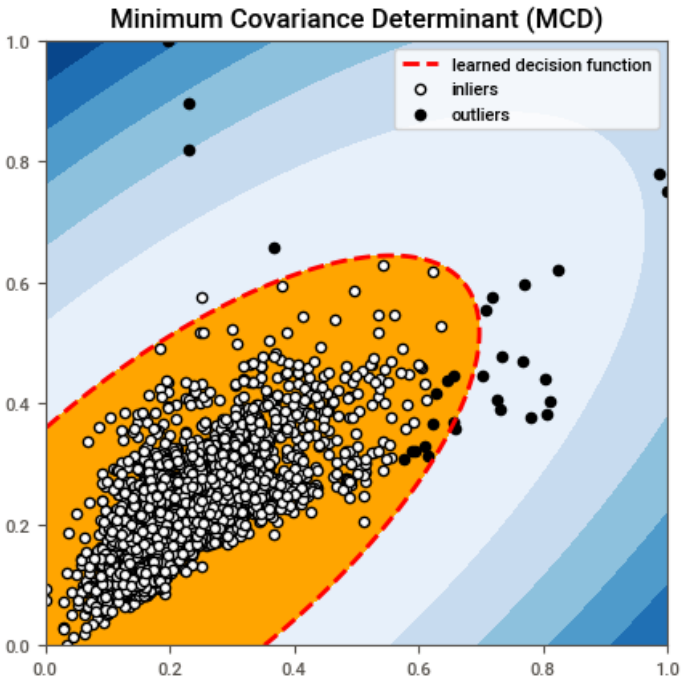

Cluster-based Local Outlier Factor (CBLOF)
learned decision function
inliers
outliers
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

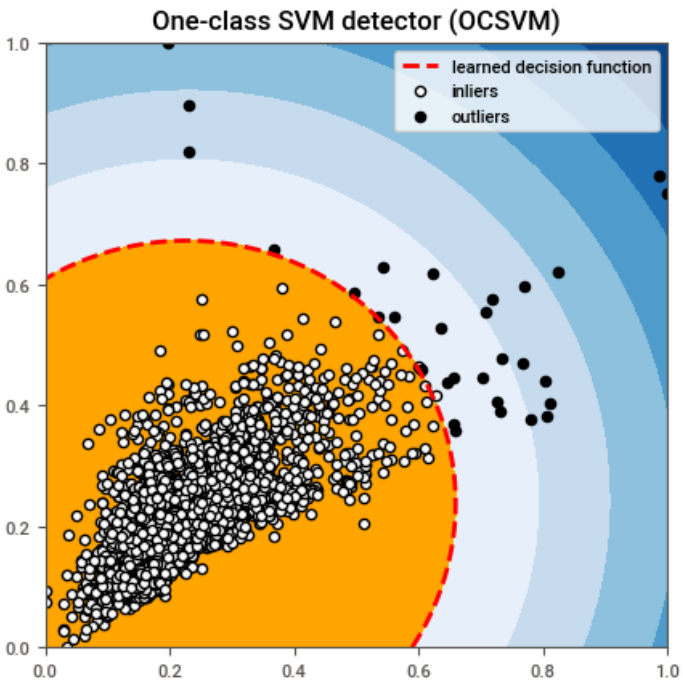

Feature Bagging
Histogram-base Outlier Detection (HBOS)
learned decision function
inliers
outliers
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Isolation Forest
learned decision function
inliers
outliers
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


K Nearest Neighbors (KNN)
learned decision function
inliers
outliers
1.0
0.8
0.6
0.4
Average KNN
learned decision function
inliers
outliers
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


PCA Outlier Detector
learned decision function


Minimum Covariance Determinant (MCD)
learned decision function
inliers
outliers
0.0
0.2
0.4
0.6
0.8
1.0


One-class SVM detector (OCSVM)
learned decision function
inliers
outliers
0.0
0.2
0.4
0.6
0.8
1.0

In [226]:
```
dfxAll
```

Out[226]:

	Gr Liv Area	SalePrice	Angle-based Outlier Detector (ABOD)	Cluster-based Local Outlier Factor (CBLOF)	Feature Bagging	Histogram-base Outlier Detection (HBOS)	Isolation Forest	K Nearest Neighbors (KNN)	Average KNN	PCA Outlier Detector	Minimu Covarian Determina (MC
0	0.249058	0.272444	0	0	0	0	0	0	0	0	
1	0.105878	0.124238	0	0	0	0	0	0	0	0	
2	0.187453	0.214509	0	0	0	0	0	0	0	0	
3	0.334589	0.311517	0	0	0	0	0	0	0	0	
4	0.243971	0.238626	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
2925	0.126036	0.174763	0	0	0	0	0	0	0	0	
2926	0.107008	0.159269	0	0	0	0	0	0	0	0	
2927	0.119819	0.160616	0	0	0	0	0	0	0	0	
2928	0.198757	0.211814	0	0	0	0	0	0	0	0	
2929	0.313866	0.236066	0	0	0	0	0	0	0	0	

2930 rows × 11 columns

6 Building the Model

In [39]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn import metrics
Model = LinearRegression()
```

In [40]:
```
results = pd.DataFrame(columns=["Method", "RMSE", "Model_Score(train)","Model_Score(test)"
```

In [42]:
```
# Let us define a function for caculating RMSE
def rmse(predictions, targets):
    return np.sqrt(((predictions - targets) ** 2).mean())
```

In [43]:
```
x_train, x_test, y_train, y_test = train_test_split(
    dfxAll[['Gr Liv Area']] , dfxAll[['SalePrice']],
    test_size=0.3, random_state=101)
Model.fit(x_train, y_train)
pred = Model.predict(x_test)
rmse_val = rmse(np.array(pred), np.array(y_test))
results.loc[0,'Method']= "Without Outlier Detection Algo"
results.loc[0,'RMSE']= rmse_val
results.loc[0,'Model_Score(train)'] = Model.score(x_train, y_train)
results.loc[0, 'Model_Score(test)'] = Model.score(x_test, y_test)
```

In [44]:
```
algo_list=dfxAll.columns[2:].to_list()
algo_list
```

Out[44]:
```
['Angle-based Outlier Detector (ABOD)',
 'Cluster-based Local Outlier Factor (CBLOF)',
 'Feature Bagging',
 'Histogram-base Outlier Detection (HBOS)',
 'Isolation Forest',
 'K Nearest Neighbors (KNN)',
 'Average KNN',
 'PCA Outlier Detector',
 'Minimum Covariance Determinant (MCD)',
 'One-class SVM detector (OCSVM)']
```

In [45]:
```
j= 1
for i in algo_list:
    x_train, x_test, y_train, y_test = train_test_split(
        dfxAll[['Gr Liv Area']][dfxAll[i]==0] , dfxAll[['SalePrice']][dfxAll[i]==0],
        test_size=0.3, random_state=101)
    Model.fit(x_train, y_train)
    pred = Model.predict(x_test)
    rmse_val = rmse(np.array(pred), np.array(y_test))
    results.loc[j,'Method']= i
    results.loc[j,'RMSE']= rmse_val
    results.loc[j,'Model_Score(train)'] = Model.score(x_train, y_train)
    results.loc[j, 'Model_Score(test)'] = Model.score(x_test, y_test)
    j= j + 1
results
```

Out[45]:

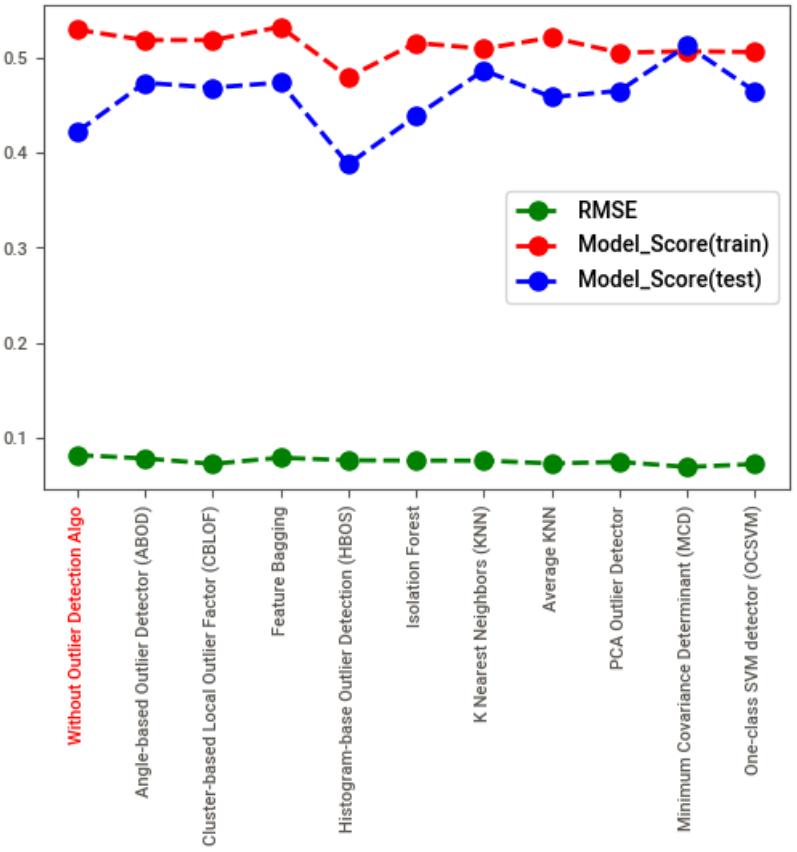
	Method	RMSE	Model_Score(train)	Model_Score(test)
0	Without Outlier Detection Algo	0.0814125	0.529892	0.422083
1	Angle-based Outlier Detector (ABOD)	0.0778883	0.519018	0.473993
2	Cluster-based Local Outlier Factor (CBLOF)	0.072126	0.518969	0.468905
3	Feature Bagging	0.0786544	0.532952	0.474194
4	Histogram-base Outlier Detection (HBOS)	0.0758688	0.479899	0.38802
5	Isolation Forest	0.0756101	0.51602	0.43892
6	K Nearest Neighbors (KNN)	0.0755176	0.509921	0.487194
7	Average KNN	0.0726374	0.52169	0.458849
8	PCA Outlier Detector	0.0742337	0.505776	0.465545
9	Minimum Covariance Determinant (MCD)	0.0688643	0.507223	0.513217
10	One-class SVM detector (OCSVM)	0.0718748	0.506361	0.464774

In [82]:
```
results[['Method', 'RMSE']]
```

Out[82]:

	Method	RMSE
0	Without Outlier Detection Algo	0.0814125
1	Angle-based Outlier Detector (ABOD)	0.0778883
2	Cluster-based Local Outlier Factor (CBLOF)	0.072126
3	Feature Bagging	0.0786544
4	Histogram-base Outlier Detection (HBOS)	0.0758688
5	Isolation Forest	0.0756101
6	K Nearest Neighbors (KNN)	0.0755176
7	Average KNN	0.0726374
8	PCA Outlier Detector	0.0742337
9	Minimum Covariance Determinant (MCD)	0.0688643
10	One-class SVM detector (OCSVM)	0.0718748

In [186]:
```
#Let us plot the results for comparison
label_list= results['Method'].tolist()
plt.plot(label_list, results['RMSE'], color='green', marker='o', linestyle='dashed',linewid
plt.plot(label_list, results['Model_Score(train)'], color='Red', marker='o', linestyle='das
plt.plot(label_list, results['Model_Score(test)'], color='Blue', marker='o', linestyle='das
plt.xticks(rotation=90)
plt.gca().get_xticklabels()[0].set_color("red")
plt.legend()
plt.show()
```



7 Conclusion

The presence of outliers in a classification or regression dataset can result in a poor fit and lower predictive modeling performance. Identifying and removing outliers is an important step before designing an ML model, veriety of algorithms should be used in the modeling pipeline and compared, to avoid model overfitting.
The obtained results in this study shows the importance of selecting the right ourlier detection algorithms while addressing the overfitting issue in our ML model. AS we can see some of the algorithm (HBOS) performed worse than our base algorithm (i.e., without outlier detection algo). Some others are doing a better job with respect to the overfitting issue.

Reference:
The present content of this notebook is highly inspired by the infromation presented in the following blog post.
https://www.analyticsvidhya.com/blog/2019/02/outlier-detection-python-pyod/
(https://www.analyticsvidhya.com/blog/2019/02/outlier-detection-python-pyod/)

In [ ]:
```
```