In [ ]:

```
import pandas as pd
import numpy as  np
import plotly as plt
import itertools
from plotly import graph_objs as go
from plotly.subplots import make_subplots
```

In [ ]:

```
#link to Train / Test data : https://www.kaggle.com/c/house-prices-advanced-regression-
techniques/data

train_data =  pd.read_csv('C:\\Users\\gopichand.g......House price Prediction\\train.cs
v')
test_data =   pd.read_csv('C:\\Users\\gopichand.g......House price Prediction\\test.cs
v')
del train_data['Id']
del test_data['Id']
train_data.head()
test_data.head()
#train_data.info()
```

In [ ]:

```
def missing_values_for_columns():
    Na_True_Fale = []
    for i in list(test_data.columns):
        Na_True_Fale.append(train_data[i].isna().any())
    return pd.DataFrame( Na_True_Fale,list(test_data.columns)).reset_index()
df = missing_values_for_columns()
```

Missing columns

In [ ]:

```
listing= []
missing_rows = []
for i in range(0,df.shape[0]):
    if df.iloc[i,1] == True:
        misssing_row = df.iloc[i,0]
        missing_rows.append(df.iloc[i,0])
        listing.append(train_data[misssing_row].isna().value_counts()[1]/train_data[mis
ssing_row].shape[0])
MissingFrame = pd.DataFrame({'Column':missing_rows,'Missing_percent':listing}).sort_val
ues('Missing_percent', ascending=False)
MissingFrame.head(5)

x  = MissingFrame['Column']
y = MissingFrame['Missing_percent']

data = go.Bar(x = x, y = y,marker=dict(color='rgba(219, 64, 82, 0.6)'), name='Missing P
ercentage')
fig = go.Figure(data = data)
fig
```

Different Neighborhoods over year

In [ ]:

```
fig = go.Figure()
for i in list(train_data['Neighborhood'].unique()):
    dat = train_data[train_data['Neighborhood']==i]
    #dat2 = dat[['YearBuilt', 'SalePrice']].sort_values('YearBuilt')
    dat2=dat[['YearBuilt', 'SalePrice']].groupby('YearBuilt').agg({'SalePrice' : 'mean'
}).sort_values('YearBuilt').reset_index()
    x= dat2['YearBuilt']
    y= dat2['SalePrice']
    fig.add_trace(go.Scatter(x = x , y = y, mode = 'lines', name = i))
fig
```

Corr Matrix

In [ ]:

```
corrmat = train_data.corr()
cor =corrmat[(corrmat > 0.5) | (corrmat  < -0.5)]
cor['Overall_relation']=cor.apply(lambda x : x.sum())
cor = cor[cor['Overall_relation']>1]
cor = cor[list(cor.index)]
cor.fillna(0, inplace = True)
fig = go.Figure(data=go.Heatmap(
                   z=cor,
                   x=list(cor.columns),
                   y=list(cor.columns), colorscale = 'Viridis'))
fig
```

In [ ]:

```
highly_correlated=list(corrmat[corrmat['SalePrice']> 0.5].index)
highly_correlated
highly_correlated_data =train_data[highly_correlated]
fig = make_subplots(4,3)
rows= list(range(4, 0, -1))
cols = list(range(3, 0, -1))
combination= list(itertools.product(rows, cols))
for i in highly_correlated:
    df_ = highly_correlated_data[[i,'SalePrice']]
    x = df_[i]
    y = df_['SalePrice']
    a = combination.pop()
    r =  a[0]
    c =  a[1]
    if i != 'SalePrice':

        fig.add_trace(go.Scatter(x = x, y = y, mode = 'markers', name = i), row = r,col
= c)
        fig.update_xaxes(title_text=i, row = r,col = c)
        fig.update_yaxes(title_text= "Sale Price", row=1, col=1)

fig.update_layout(height=1000, width=1000,
                  title_text="High Correlated variable VS Sale Price")

fig
```

Target Variable Log Transformed (As its is right Skewed)

In [ ]:

```
#x  = train_data['SalePrice']
x=np.log(train_data["SalePrice"])
fig = go.Figure(data=[go.Histogram(x=x)])
fig
```

Outliers Detection

In [448]:

```
 # Yet to come :)
```

In [ ]:

```

```