In [2]:
```
import pandas as pd
import missingno as msno
```

In [ ]:
```
df.isnull().sum()     # check null values
```

In [ ]:
```
df.dtypes # shows columns of which datadype
```

In [ ]:
```
df = df.rename(columns={'"footfall':'footfall','fail"':'fail'})
df['fail'] = df['fail'].map({'1"':1,'0"':0})
final_df['duration_min'].replace(to_replace =np.nan,value ="0")
df.where(df > 0.5, other=np.nan)
```

In [ ]:
```
dataframe.at[index,'column-name']='new value'
dataframe.loc[row index,['column-names']] = value
```

In [ ]:
```
df1["Fabric"].value_counts()
```

In [ ]:
```
df['fail'].unique()  # shows unique values of column values
```

In [ ]:
```
df['footfall']=df['footfall'].astype(int)    # change datatype to int
```

In [ ]:
```
msno.bar(df)
plt.show         # shows missing values of columns
```

In [ ]:
```
df.groupby('fail').count()    # shows groupby a specific column
df.groupby(['campaign','y']).size().reset_index()
df.groupby(['contact','y']).count()
df.groupby(['publication', 'date_m']).agg(['mean', 'count', 'sum'])
```

In [ ]:
```
df.filter(items=['a', 'b']) # by col
```

In [ ]:
```
df[(df['Survived']==0)&(df['Age']<40)
```

In [ ]:
```
df.hist()    # shows histograms of columns
df.plot.hist(bins=25, alpha=0.5)
df['A'].plot.hist(bins=20)
```

In [ ]:
```
df.corr()['target'] # correlation with target with all columns
df.corr() # pairwise correlation cols
df.cov() # pairwise covariance cols
df.kurt() # kurtosis over cols (def)
df.mad() # mean absolute deviation
df.sem() # standard error of mean
df.var() # variance over cols (def)
```

In [ ]:
```
df.drop(123,axis=0,inplace=True)
df1.drop(['day','month','year'],axis=1)
```

In [ ]:
```
X = df.drop(columns='fail',axis=1) # drop columns
y=df['fail']
```

In [ ]:
```
X=df.iloc[:,1:13]
y=df.iloc[:,-1]
```

In [ ]:
```
df2.loc[df2["Product Name"]=='Name N/A',: ]
```

In [ ]:
```
df = pd.merge(df1, df2, on="PID")
```

In [ ]:
```
df[df['y']=='yes']
```

In [ ]:
```
for i,j in zip(age_sum['index'],age_sum['age']):
    percentage=np.round((j/df_yes.shape[0])*100,1)
    print([i,percentage],end='')
```

In [ ]:
```
pd.read_csv('file.csv',header=None)  === takes a dataframe without any column name
pd.read_csv('file.csv',names=['a','b','c','d']) === gives column names while taking data
df=pd.read_csv(r'C:/Users/pbann/Downloads/data.csv',sep='\,')
pd.read_csv('file.csv',sep = '@')  === if data given is seperated by @
pd.read_csv('file.csv',sep = '@',skiprows=[1,3]) === removes rows which we dont need
pd.read_excel('file.xlsx',sheet_name = 'sheet1') === gives selected sheet in excel file
df = pd.Excelfile('file.xlsx') ---- df.sheet_names == gives sheets available in excel file
for i in df.sheet_names:
        pd.read_excel('file.xlsx',i)      read all sheets one by one
df.to_csv('pk.csv',index=False,header = False)   === to store in diff formats with no index and column
names
df = pd.read_csv('file.csv', header=0,index_col=0, quotechar='"', sep=':',na_values = ['na', '-', '.',
''])
```

In [ ]:
```
import requests           # to get raw data
    url =
    data = requests.get(url)
    data1 = data.json()
    pd.dataframe(data1,colums=['1','2','22'])
```

In [ ]:
```
df1.to_csv('final_df1.csv')
```

In [ ]:
```
df.columns == shows all columns
```

In [ ]:
```
df[['column1','column2']] = shows only thats selected column  gives dataframe
df['column1'] == for single column with single bracket = series , double brackets = dataframe
df[['column1']] == dataframe
```

In [ ]:
```
df2 = pd.DataFrame(df.Cabin.str.split('([a-zA-Z]+)[^a-zA-Z]+)'))
    or
    df['Cabin_Number'] = df['Cabin'].str.replace('([a-zA-Z]+)','')
    df['Cabin_letter'] = df['Cabin'].str.extract('([a-zA-Z]+)')
```

In [ ]:
```
df['name'].str.startswith('s')['Name']
```

In [ ]:
```
df2['col4_4'] = df['col1'].apply(test)
def test(a):
    return a**2
```

In [ ]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['job']= label_encoder.fit_transform(df['job'])
```

In [ ]:
```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder()
final_df['Airline']=ohe.fit_transform(final_df['Airline'])
```

In [ ]:
```
pd.get_dummies(df['Gender'])
```

In [ ]:
```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
x = df1[['Temperature','RH','Ws','Rain','FFMC','FWI']]
y=df1[["classes"]]
X = sm.add_constant(x)
vif_data = pd.DataFrame()
vif_data["feature"] = x.columns
vif_data["VIF"] = [variance_inflation_factor(x.values, i) for i in range (len(x.columns))]
vif_data
```

In [ ]:
```
df['Product_Category_2']=df['Product_Category_2'].fillna(df['Product_Category_2'].mode()[0])
```

In [ ]:
```
df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.replace('+','')
final_df['duration_hour']=final_df['Duration'].str.split(' ').str[0].str.split('h').str[0]
```

In [ ]:
```
final_df=train_df.append(test_df)
```

In [ ]:
```
df = df.copy() # copy a DataFrame
df = df.rank() # rank each col (default)
```

In [ ]:
```
df.to_string() # to string
df.as_matrix() # to numpy matrix
```

In [ ]:
```
np.random.normal(0,1,999)
```

In [ ]:
```
df1 = df.cumsum()
```

In [ ]:
```
bins = np.linspace(-10,15,26)
```

In [ ]:
```
ax = df.plot.kde()
# followed by the standard plot code as above
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: