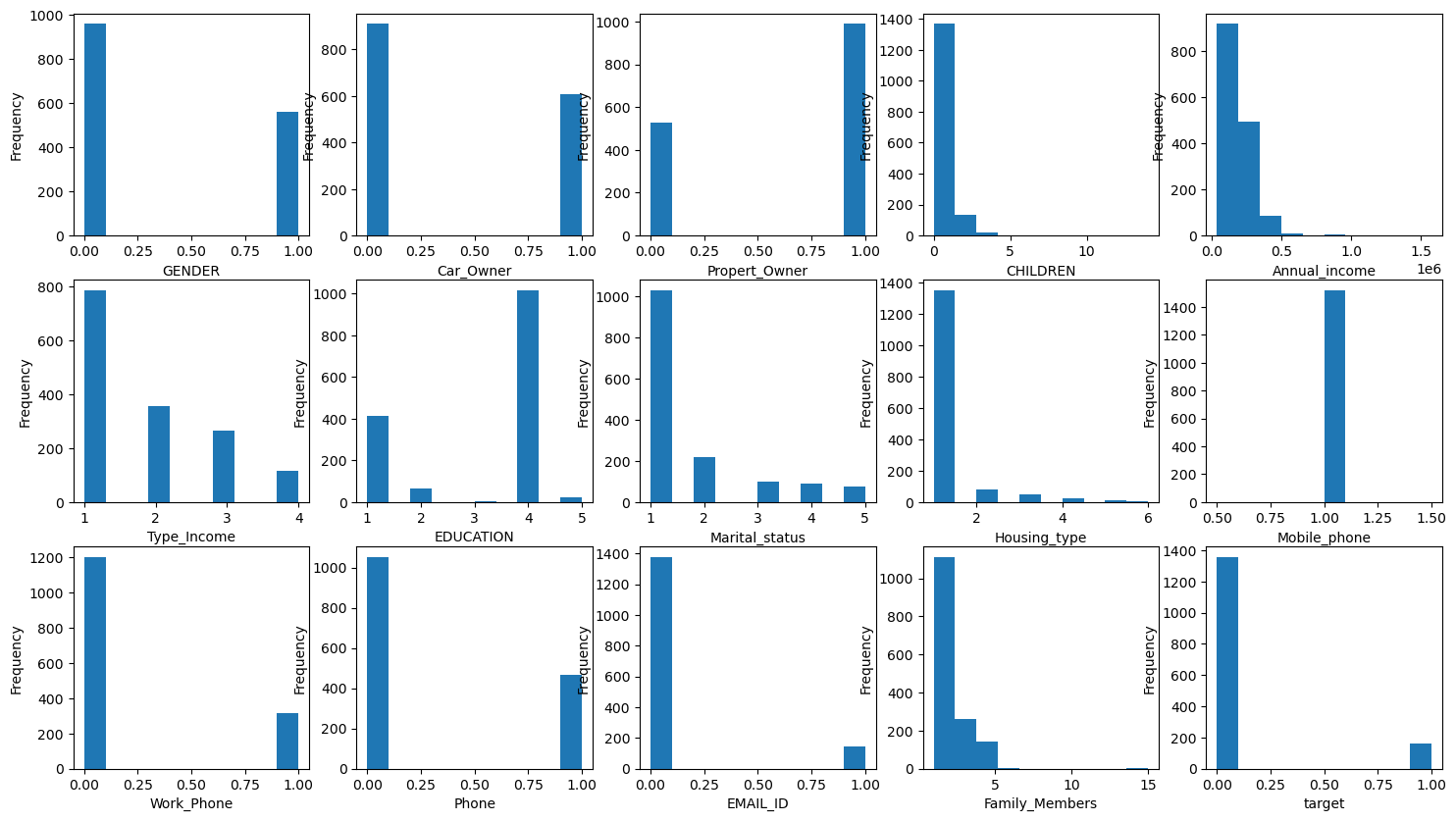
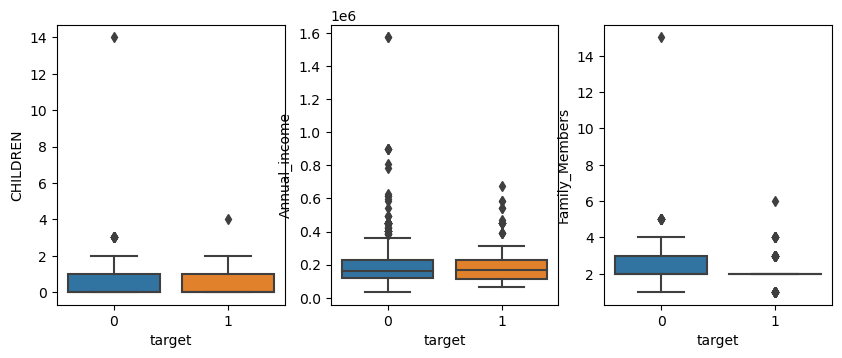
1. Observe all attributes



1. Observe the numerical elements for outliers



1. Perform ANOVA test on numerical attributes

Code:

from sklearn.feature\_selection import SelectKBest, f\_classif, chi2

x = df.loc[:, ['CHILDREN', 'Annual\_income', 'Family\_Members']]

y = df.loc[:, 'target']

fs = SelectKBest(score\_func=f\_classif, k='all') # call the method

bestFeatures = fs.fit(x, y) # train the model

np.set\_printoptions(suppress = True)

print(bestFeatures.scores\_) # print out the scores

print(bestFeatures.pvalues\_)

Result:

[0.59435181 0.49419043 1.27207939]

[0.44086135 0.48217192 0.25955542]

4:Perform Chi-squared test on categorical attributes

Code:

x = df.loc[:, ['GENDER','Car\_Owner', 'Propert\_Owner', 'Type\_Income','EDUCATION', 'Marital\_status']]

y = df.loc[:, 'target']

chi = SelectKBest(score\_func=chi2, k='all')

catFeatures = chi.fit(x, y)

print(catFeatures.scores\_)

print(catFeatures.pvalues\_)

Result:

[2.85930153 0.41177612 0.13332647 0.09766221 0.0225236 0.02593438]

[0.09084694 0.52106939 0.71500767 0.75465332 0.88070256 0.87206068]

Code:

x = df.loc[:, ['Housing\_type', 'Mobile\_phone','Work\_Phone', 'Phone', 'EMAIL\_ID']]

y = df.loc[:, 'target']

chi = SelectKBest(score\_func=chi2, k='all')

catFeatures = chi.fit(x, y)

print(catFeatures.scores\_)

print(catFeatures.pvalues\_)

Result:

[5.97336925 0. 0.22799915 0.16791828 0.06750606]

[0.01452351 1. 0.63301142 0.68196856 0.79500317]

5:Turn categorical variables into dummy variables

Code:

x = df\_withdummies.loc[:, df\_withdummies.columns!='target']

y = df\_withdummies['target']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=0, stratify=y)

6:Balance The data

Code:

from imblearn.over\_sampling import SMOTE

os = SMOTE(random\_state=0) # call the method.

oversampled\_x,oversampled\_y=os.fit\_resample(x\_train, y\_train)

print(x\_train.shape)

print(oversampled\_x.shape)

Result:

(1062, 6)

(1898, 6)

7:Build the prediction model

Code:

LogRegression = LogisticRegression(penalty=None, max\_iter=2000)

LogRegression.fit(oversampled\_x, oversampled\_y.values.ravel())

8:Evaluate the model

Code:

from sklearn.metrics import accuracy\_score

test\_pred = LogRegression.predict(x\_test)

accuracy\_score(y\_test, test\_pred)

Result:

0.6776315789473685