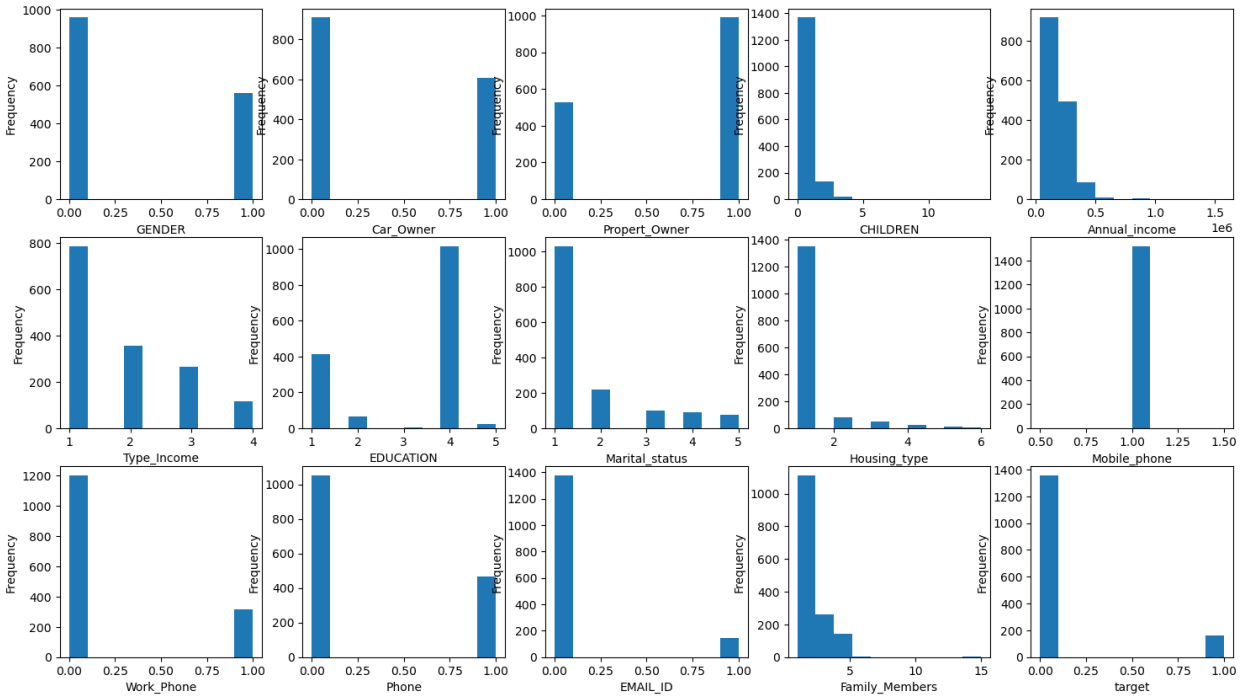
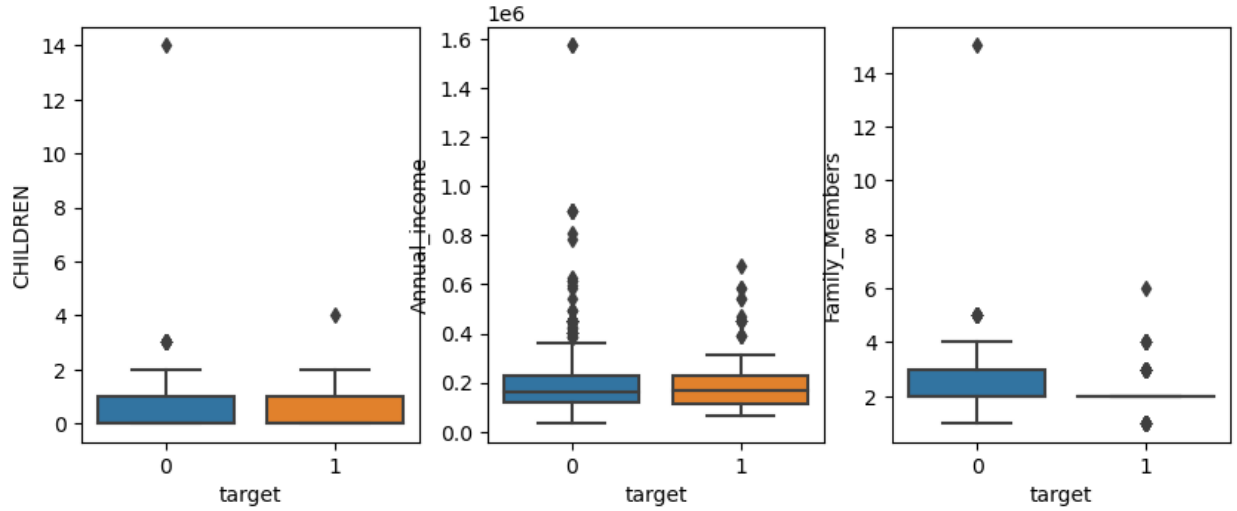
1. Observe all attributes



2. Observe the numerical elements for outliers



3. 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
```