Anexe

A. Grafice

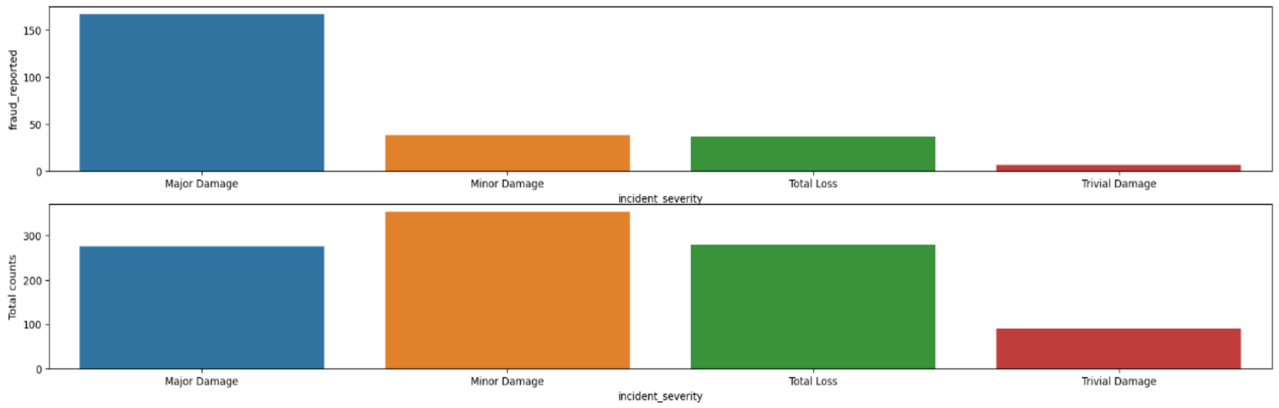


Figure 1. Grafic fraud_reported & incident_severity

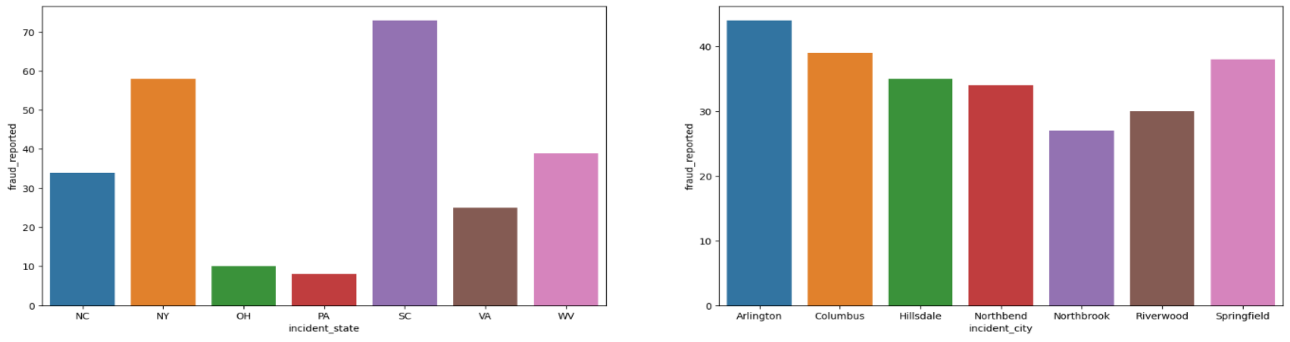


Figure 2. Grafic fraud_reported & incident_state, incident_city

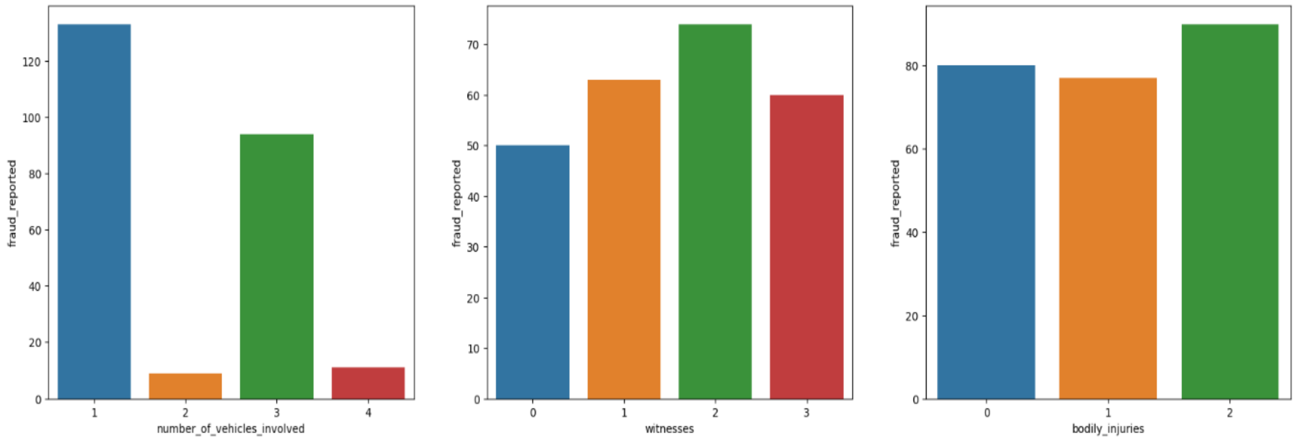
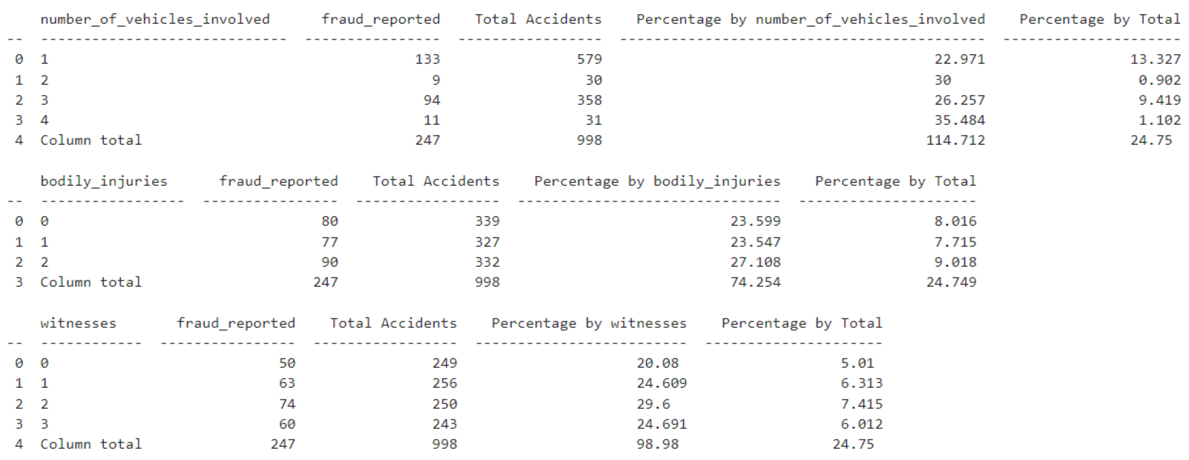


Figure 3. Grafic fraud_reported & number_of_vehicles_involved, witnesses, bodily_injuries



```
   number_of_vehicles_involved      fraud_reported    Total Accidents    Percentage by number_of_vehicles_involved    Percentage by Total
-- -----------------------------  ----------------  -----------------  -------------------------------------------  ---------------------
 0  1                                          133                579                                       22.971                 13.327
 1  2                                            9                 30                                       30                      0.902
 2  3                                           94                358                                       26.257                  9.419
 3  4                                           11                 31                                       35.484                  1.102
 4  Column total                               247                998                                      114.712                 24.75

   bodily_injuries     fraud_reported    Total Accidents    Percentage by bodily_injuries    Percentage by Total
-- -----------------  ----------------  -----------------  -------------------------------  ---------------------
 0  0                                80                339                           23.599                  8.016
 1  1                                77                327                           23.547                  7.715
 2  2                                90                332                           27.108                  9.018
 3  Column total                    247                998                           74.254                 24.749

   witnesses      fraud_reported    Total Accidents    Percentage by witnesses    Percentage by Total
-- ------------  ----------------  -----------------  -------------------------  ---------------------
 0  0                          50                249                      20.08                   5.01
 1  1                          63                256                      24.609                  6.313
 2  2                          74                250                      29.6                    7.415
 3  3                          60                243                      24.691                  6.012
 4  Column total              247                998                      98.98                  24.75
```

Figure 4. Concordanță Figure 3

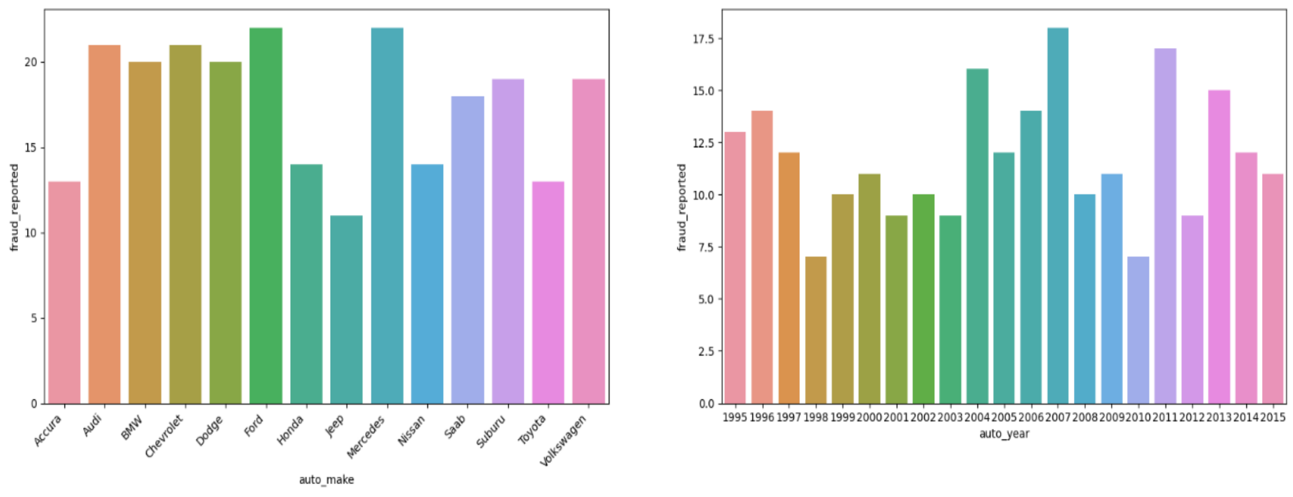


Figure 5. Grafic fraud_reported & auto_make, auto_year

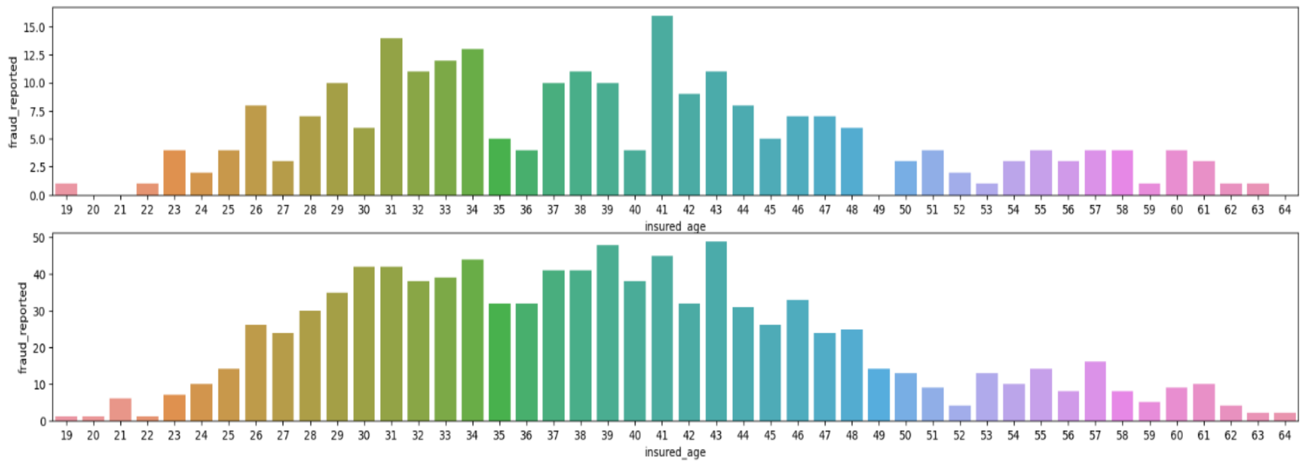


Figure 6. Grafic fraud_reported & insured_age

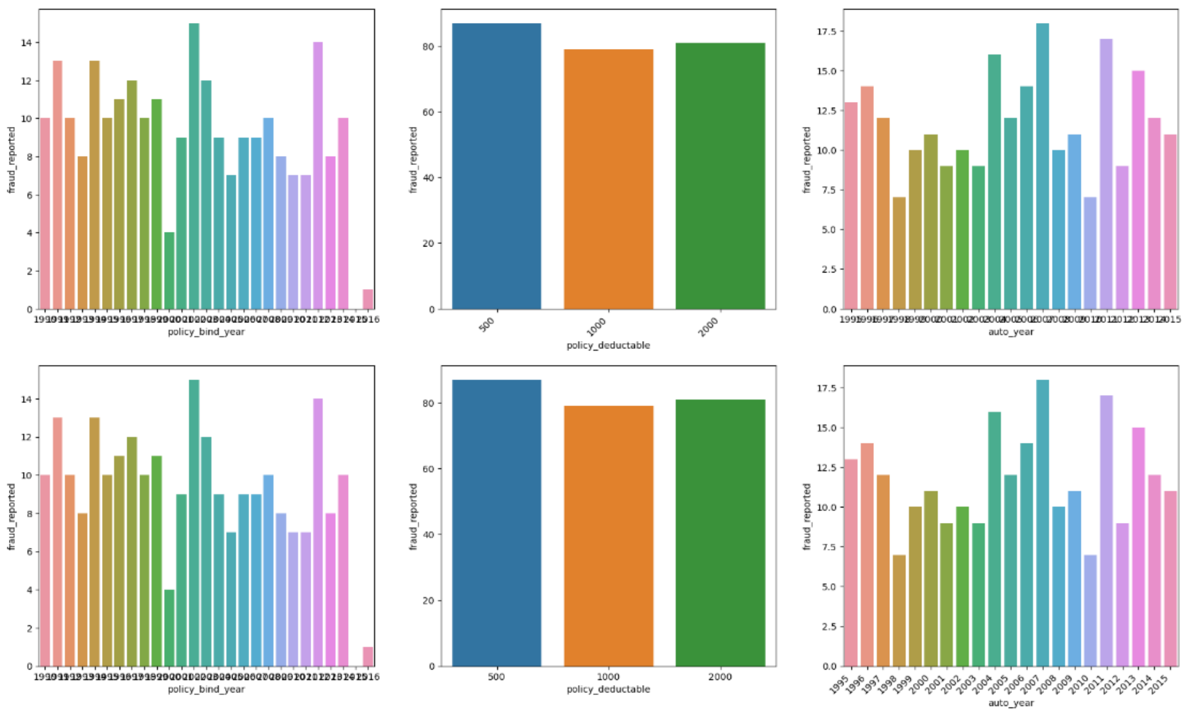


Figure 7. Grafic fraud_reported & policy_bind_year, policy_deductable, auto_year

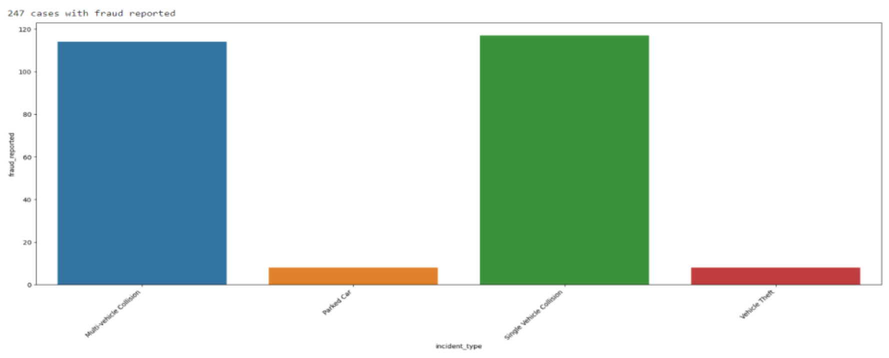


Figure 8. Grafic fraud_reported & incident_type

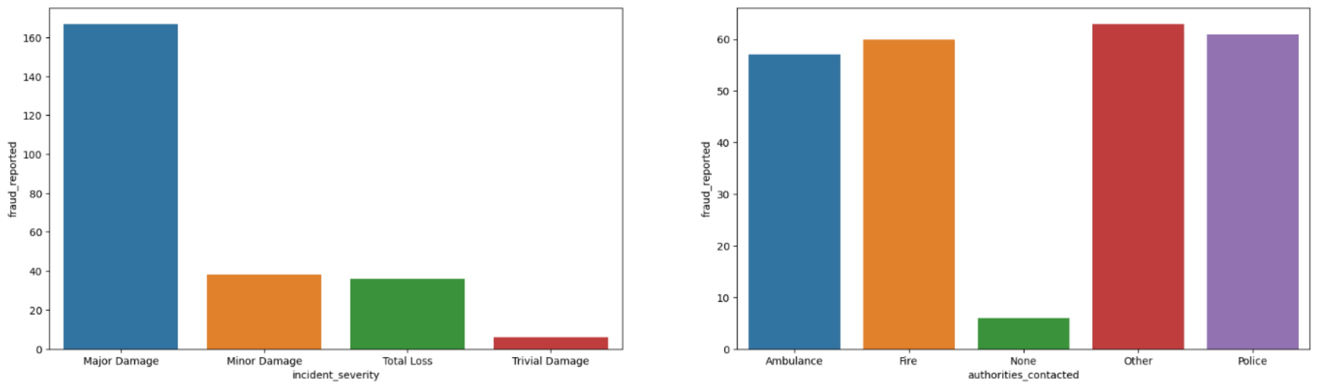


Figure 9. Grafic fraud_reported & incident_severity, authorities_contacted

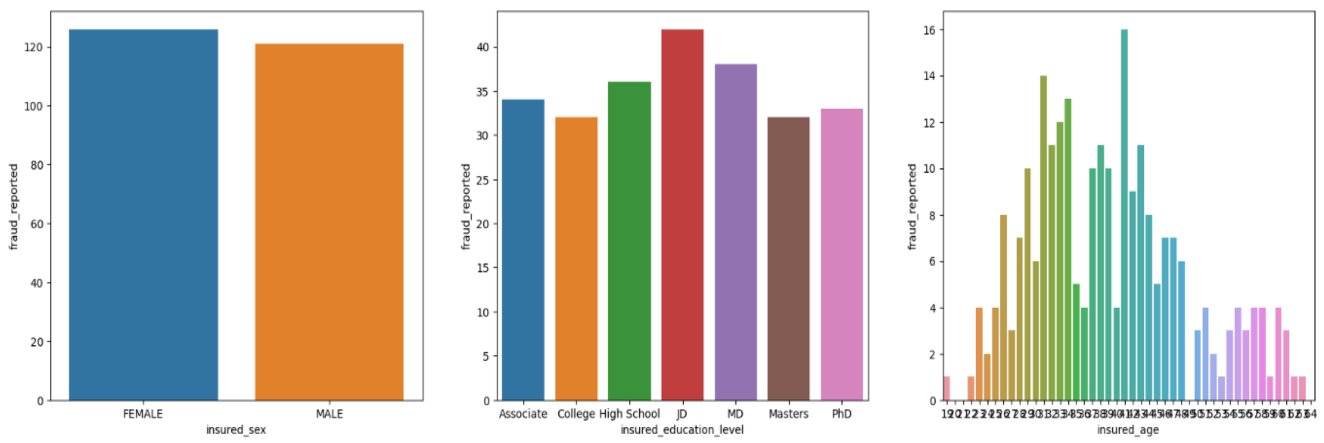


Figure 10. Grafic fraud_reported & insured_sex, insured_education_level, insured_age

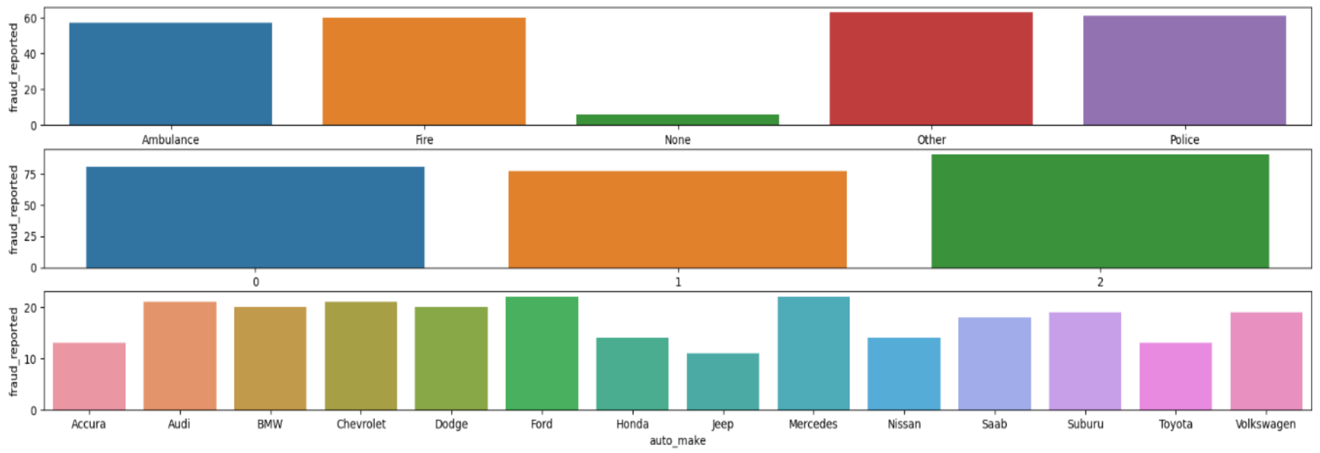


Figure 11. Grafic fraud_reported & authorities_contacted, bodily_injuries, auto_make

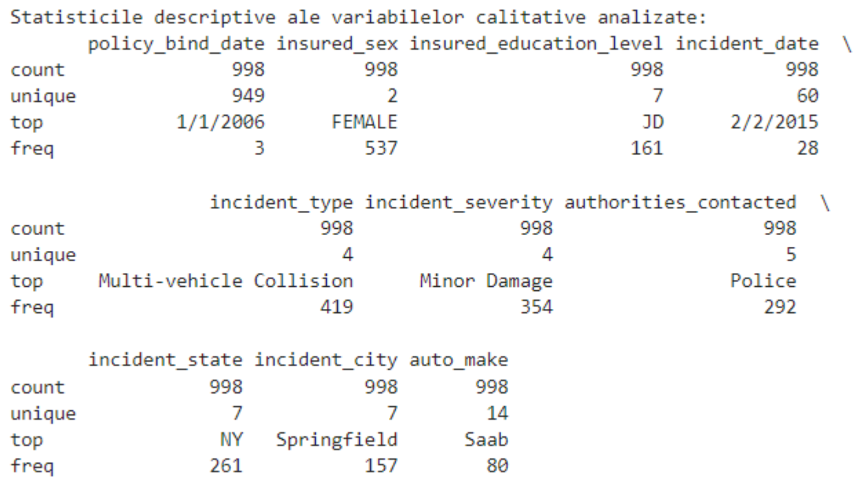
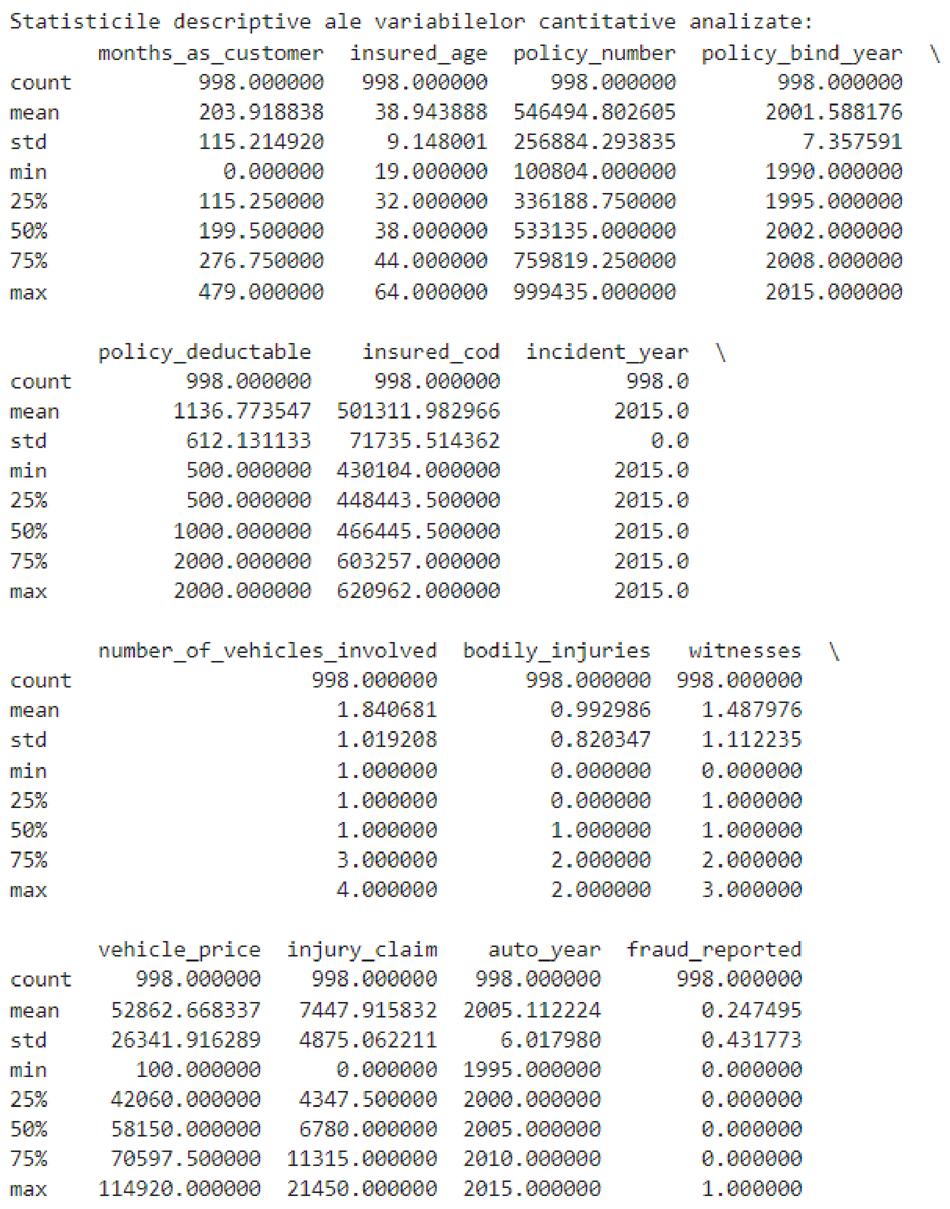
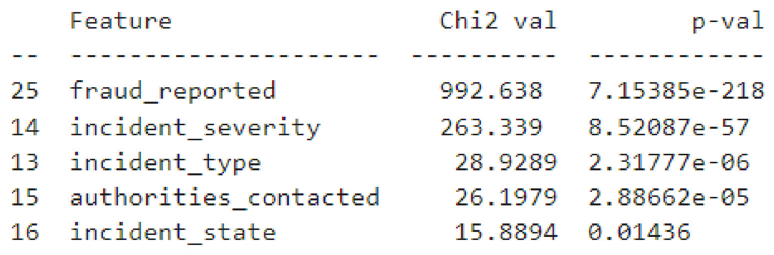

```
Statisticile descriptive ale variabilelor calitative analizate:
       policy_bind_date insured_sex insured_education_level incident_date  \
count               998         998                     998           998
unique              949           2                       7            60
top            1/1/2006      FEMALE                      JD      2/2/2015
freq                  3         537                     161            28

                  incident_type incident_severity authorities_contacted  \
count                       998               998                   998
unique                        4                 4                     5
top      Multi-vehicle Collision      Minor Damage                Police
freq                        419               354                   292

       incident_state incident_city auto_make
count             998           998       998
unique              7             7        14
top                NY   Springfield      Saab
freq              261           157        80
```

Figure 12. Statistici descriptive ale variabilelor calitative



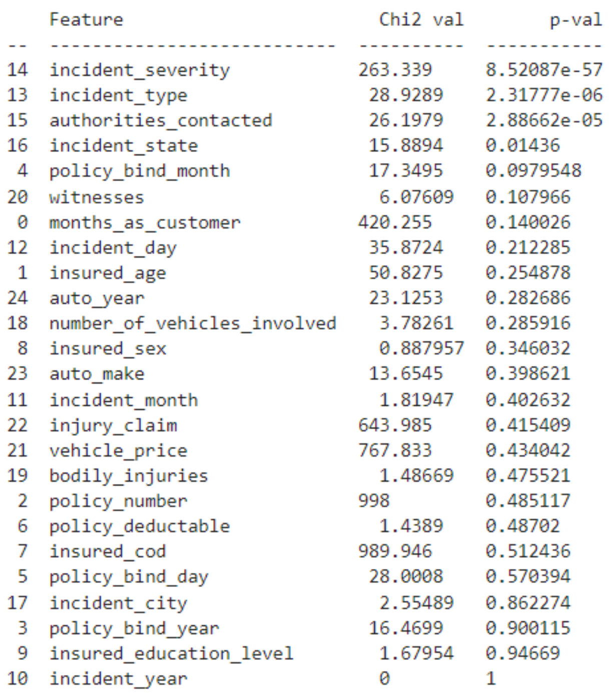
```
Statisticile descriptive ale variabilelor cantitative analizate:
       months_as_customer  insured_age  policy_number  policy_bind_year  \
count          998.000000   998.000000     998.000000        998.000000   
mean           203.918838    38.943888  546494.802605       2001.588176   
std            115.214920     9.148001  256884.293835          7.357591   
min              0.000000    19.000000  100804.000000       1990.000000   
25%            115.250000    32.000000  336188.750000       1995.000000   
50%            199.500000    38.000000  533135.000000       2002.000000   
75%            276.750000    44.000000  759819.250000       2008.000000   
max            479.000000    64.000000  999435.000000       2015.000000   

       policy_deductable    insured_cod  incident_year  \
count         998.000000     998.000000          998.0   
mean         1136.773547  501311.982966         2015.0   
std           612.131133   71735.514362            0.0   
min           500.000000  430104.000000         2015.0   
25%           500.000000  448443.500000         2015.0   
50%          1000.000000  466445.500000         2015.0   
75%          2000.000000  603257.000000         2015.0   
max          2000.000000  620962.000000         2015.0   

       number_of_vehicles_involved  bodily_injuries   witnesses  \
count                   998.000000       998.000000  998.000000   
mean                      1.840681         0.992986    1.487976   
std                       1.019208         0.820347    1.112235   
min                       1.000000         0.000000    0.000000   
25%                       1.000000         0.000000    1.000000   
50%                       1.000000         1.000000    1.000000   
75%                       3.000000         2.000000    2.000000   
max                       4.000000         2.000000    3.000000   

       vehicle_price  injury_claim    auto_year  fraud_reported  
count     998.000000    998.000000   998.000000      998.000000  
mean    52862.668337   7447.915832  2005.112224        0.247495  
std     26341.916289   4875.062211     6.017980        0.431773  
min       100.000000      0.000000  1995.000000        0.000000  
25%     42060.000000   4347.500000  2000.000000        0.000000  
50%     58150.000000   6780.000000  2005.000000        0.000000  
75%     70597.500000  11315.000000  2010.000000        0.000000  
max    114920.000000  21450.000000  2015.000000        1.000000  
```

Figure 13. Statistici descriptive pentru variabilele cantitative



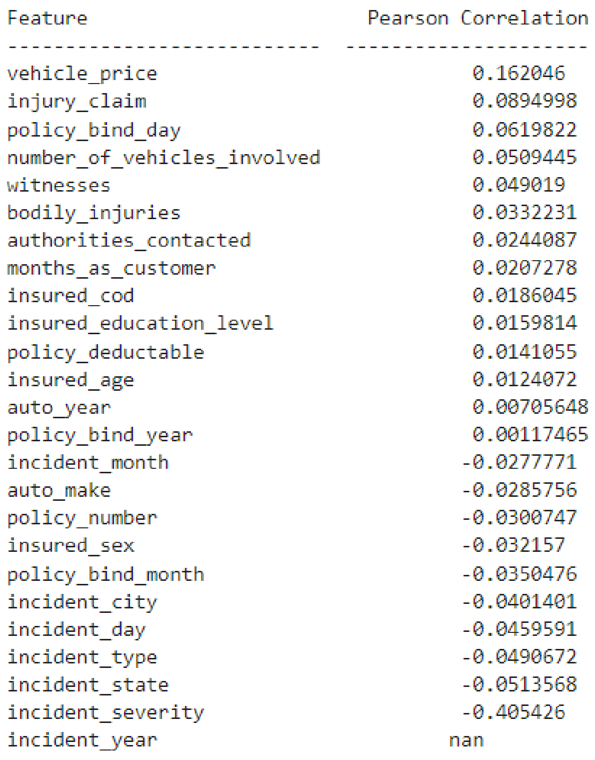
```
    Feature                   Chi2 val         p-val
--  ---------------------  ----------  ------------
25  fraud_reported            992.638   7.15385e-218
14  incident_severity         263.339   8.52087e-57
13  incident_type              28.9289  2.31777e-06
15  authorities_contacted      26.1979  2.88662e-05
16  incident_state             15.8894  0.01436
```

Figure 14.Testul Chi2, variabile semnificative



```
    Feature                        Chi2 val        p-val
--  --------------------------  ----------  -----------
14  incident_severity            263.339     8.52087e-57
13  incident_type                 28.9289    2.31777e-06
15  authorities_contacted         26.1979    2.88662e-05
16  incident_state                15.8894    0.01436
 4  policy_bind_month             17.3495    0.0979548
20  witnesses                      6.07609   0.107966
 0  months_as_customer           420.255     0.140026
12  incident_day                  35.8724    0.212285
 1  insured_age                   50.8275    0.254878
24  auto_year                     23.1253    0.282686
18  number_of_vehicles_involved    3.78261   0.285916
 8  insured_sex                    0.887957  0.346032
23  auto_make                     13.6545    0.398621
11  incident_month                 1.81947   0.402632
22  injury_claim                 643.985     0.415409
21  vehicle_price                767.833     0.434042
19  bodily_injuries                1.48669   0.475521
 2  policy_number                998         0.485117
 6  policy_deductable              1.4389    0.48702
 7  insured_cod                  989.946     0.512436
 5  policy_bind_day               28.0008    0.570394
17  incident_city                  2.55489   0.862274
 3  policy_bind_year              16.4699    0.900115
 9  insured_education_level        1.67954   0.94669
10  incident_year                  0         1
```

Figure 15. Testul Chi2 după transformarea variabilelor calitative în variabile cantitative



```
Feature                          Pearson Correlation
---------------------------  ---------------------
vehicle_price                          0.162046
injury_claim                           0.0894998
policy_bind_day                        0.0619822
number_of_vehicles_involved            0.0509445
witnesses                              0.049019
bodily_injuries                        0.0332231
authorities_contacted                  0.0244087
months_as_customer                     0.0207278
insured_cod                            0.0186045
insured_education_level                0.0159814
policy_deductable                      0.0141055
insured_age                            0.0124072
auto_year                              0.00705648
policy_bind_year                       0.00117465
incident_month                        -0.0277771
auto_make                             -0.0285756
policy_number                         -0.0300747
insured_sex                           -0.032157
policy_bind_month                     -0.0350476
incident_city                         -0.0401401
incident_day                          -0.0459591
incident_type                         -0.0490672
incident_state                        -0.0513568
incident_severity                     -0.405426
incident_year                        nan
```

Figure 16. Coeficientul de corelație Pearson

B. Script Python

```
#Instalarea pachetelor si importul librariilor ce vor fi utilizate

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%pip install seaborn

import seaborn as sns

import time

%pip install pandasql

from pandasql import sqldf

%pip install imblearn

from imblearn.over_sampling import SMOTE

import scipy.stats as stats

%pip install tabulate

from tabulate import tabulate

%pip install xgboost

from xgboost import XGBClassifier

from xgboost import plot_tree

from xgboost import plot_importance

%pip install hyperopt

from hyperopt import STATUS_OK, Trials, fmin, hp, tpe

import hyperopt.pyll

from hyperopt.pyll import scope

%pip install hpsklearn

from hpsklearn import HyperoptEstimator
```

```
%pip install sklearn

from sklearn.tree import DecisionTreeClassifier

from sklearn.tree import plot_tree

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.model_selection import train_test_split

from sklearn.model_selection import cross_val_score

from sklearn.model_selection import cross_validate

from sklearn.model_selection import RepeatedStratifiedKFold

from sklearn.model_selection import GridSearchCV

from sklearn import metrics

%pip install category_encoders

from category_encoders.ordinal import OrdinalEncoder

from category_encoders.binary import BinaryEncoder

from category_encoders.one_hot import OneHotEncoder

from IPython.display import display

#Importarea setului de date

df = pd.read_csv("pjSda.csv")

print('The data contains ', len(df),' observations.')

print(df.head())

#Prezentarea tipului datelor utilizate

print(df.dtypes)

#Valorile unice

print('months_as_customer',df['months_as_customer'].unique())

print('insured_age',df['insured_age'].unique())
print('policy_number',df['policy_number'].unique())

print('policy_bind_date',df['policy_bind_date'].unique())

print('policy_bind_year',df['policy_bind_year'].unique())

print('policy_deductable',df['policy_deductable'].unique())

print('insured_cod',df['insured_cod'].unique())

print('insured_sex',df['insured_sex'].unique())

print('insured_education_level',df['insured_education_level'].unique())

print('incident_date',df['incident_date'].unique())

print('incident_year',df['incident_year'].unique())

print('incident_type',df['incident_type'].unique())

print('incident_severity',df['incident_severity'].unique())

print('authorities_contacted',df['authorities_contacted'].unique())

print('incident_state',df['incident_state'].unique())

print('incident_city',df['incident_city'].unique())

print('number_of_vehicles_involved',df['number_of_vehicles_involved'].unique())

print('bodily_injuries',df['bodily_injuries'].unique())

print('witnesses',df['witnesses'].unique())

print('vehicle_price',df['vehicle_price'].unique())

print('injury_claim', df['injury_claim'].unique())

print('auto_make',df['auto_make'].unique())

print('auto_year',df['auto_year'].unique())

print('fraud_reported',df['fraud_reported'].unique())

#Investigarea valorilor nule in insured_education_level, vehicle_price, auto_year, number_of_vehicles_involved

print('insured_education_level has ', len(df.loc[(df['insured_education_level']=='NA')]), ' row(s) with a NA')
print('vehicle_price has ',len(df.loc[(df['vehicle_price']=='NA')]),' row(s) with a NA')

print('auto_year has ',len(df.loc[(df['auto_year']=='NA')]),' row(s) with a NA')

print('number_of_vehicles_involved has
',len(df.loc[(df['number_of_vehicles_involved']=='NA')]),' row(s) with a NA')

print(' ')

print(df.loc[(df['insured_education_level']==' NA')])

print(df.loc[(df['vehicle_price']=='NA')])

print(df.loc[(df['auto_make']=='NA')])

print(df.loc[(df['number_of_vehicles_involved']=='NA')])

#Renuntam la randurile 916 si 990, creand un nou data frame

df2 = df.loc[df['insured_education_level']!='NA']

#Pentru coloana Vehicle price, randul 13 com inlocui valoarea lipsa cu media preturilor
autovehiculelor

df2_vehicle_price_idx = (df2['vehicle_price']==0)

df2.loc[list(df2_vehicle_price_idx),'df2_vehicle_price_idx']=52863

#Verificam daca exista dubluri

print(len(df2.drop_duplicates())==len(df2))

#Statistici descriptive

print('Statisticile descriptive ale variabilelor cantitative analizate:')

print (df2.describe())

print('Statisticile descriptive ale variabilelor calitative analizate:')

print (df2.describe(include=['object']))

#Relatia dintre variabile - testul chi2

df2_chi_result = []

for feat in df2.columns:
    chi2_val, p_val, dof2, ex1 = stats.chi2_contingency(pd.crosstab(df2[feat], df2['fraud_reported']))

    df2_chi_result.append([feat, chi2_val, p_val])

chi_df = pd.DataFrame(df2_chi_result, columns=['Features', 'Chi2 val', 'p-val'])

chi_df.sort_values(by='p-val', ascending=True, inplace=True)

#Pastram doar variabilele relevante, cu un p-val < 0.05

print(tabulate(chi_df[chi_df['p-val'] < 0.05], headers=['Feature', 'Chi2 val', 'p-val']))

#Grafic - fraud_reported & incident_severity

gpd_val1=df2.groupby('incident_severity').agg({'fraud_reported':'sum'}).reset_index()

gpd_val2=df2.groupby('incident_severity').agg('count').reset_index()

fig, (ax1,ax2) = plt.subplots(2,1,figsize=(22, 6))

sns.barplot(x='incident_severity', y='fraud_reported', data = gpd_val1, ax=ax1)

sns.barplot(x='incident_severity', y='fraud_reported', data=gpd_val2, ax=ax2)

ax2.set(ylabel='Total counts')

plt.show()

total_list = pd.concat([gpd_val1, gpd_val2['fraud_reported'].rename('Total Accidents')],axis=1)

total_list = pd.concat([gpd_val1, gpd_val2['fraud_reported'].rename('Total Accidents')],axis=1)

total_list['Percentage by incident_severity']= round((total_list['fraud_reported']/total_list['Total Accidents'])*100,3)

total_list['Percentage by Total'] = round((total_list['fraud_reported']/sum(total_list['Total Accidents']))*100,3)

ax2.set(ylabel='Total counts')

plt.show()

#Grafic - fraud_reported & incident_state,incident_city

gpd_val3=df2.groupby('incident_state').agg({'fraud_reported':'sum'}).reset_index()
```

```
gpd_val4=df2.groupby('incident_state').agg('count').reset_index()

gpd_val5=df2.groupby('incident_city').agg({'fraud_reported':'sum'}).reset_index()

gpd_val6=df2.groupby('incident_city').agg('count').reset_index()

fig, (ax1, ax3) = plt.subplots(1,2,figsize=(22, 6))

sns.barplot(x='incident_state', y='fraud_reported', data = gpd_val3, ax=ax1)

sns.barplot(x='incident_state', y='fraud_reported', data = gpd_val4, ax=ax2)

sns.barplot(x='incident_city', y='fraud_reported', data = gpd_val5, ax=ax3)

sns.barplot(x='incident_city', y='fraud_reported', data = gpd_val6, ax=ax4)

plt.show()

total_list1 = pd.concat([gpd_val3, gpd_val4['fraud_reported'].rename('Total Accidents')],axis=1)

total_list1['Percentage by Incident state']= round((total_list1['fraud_reported']/total_list1['Total Accidents'])*100,3)

total_list1['Percentage by Total'] = round((total_list1['fraud_reported']/sum(total_list1['Total Accidents']))*100,3)

total_list2 = pd.concat([gpd_val5, gpd_val6['fraud_reported'].rename('Total Accidents')],axis=1)

total_list2['Percentage by Incident city']= round((total_list2['fraud_reported']/total_list2['Total Accidents'])*100,3)

total_list2['Percentage by Total'] = round((total_list2['fraud_reported']/sum(total_list2['Total Accidents']))*100,3)

data1 = [['Column total'],

[sum(total_list1['fraud_reported'])],

[sum(total_list1['Total Accidents'])],

[sum(total_list1['Percentage by Incident state'])],

[sum(total_list1['Percentage by Total'])]]

data2 = [['Column total'],
[sum(total_list2['fraud_reported'])],
[sum(total_list2['Total Accidents'])],
[sum(total_list2['Percentage by Incident city'])],
[sum(total_list2['Percentage by Total'])]]
nr1 = pd.DataFrame(data1)
nr1 = nr1.transpose()
nr1.rename(columns={0:'incident_state',1:'fraud_reported',2:'Total Accidents',3:'Percentage by Incident state',4:'Percentage by Total'}, inplace=True)
tl1=pd.concat([total_list1,nr1],ignore_index=True)
nr2 = pd.DataFrame(data2)
nr2 = nr2.transpose()
nr2.rename(columns={0:'incident_city',1:'fraud_reported',2:'Total Accidents',3:'Percentage by Incident city',4:'Percentage by Total'}, inplace=True)
tl2=pd.concat([total_list2,nr2],ignore_index=True)
print(tabulate(tl1, headers=tl1.columns))
print(' ')
print(tabulate(tl2, headers=tl2.columns))
#Grafic - fraud_reported & number_of_vehicles_involved,bodily_injuries,witnesses
gpd_val1 = df2.groupby('number_of_vehicles_involved').agg({'fraud_reported': 'sum'}).reset_index()
gpd_val2 = df2.groupby('bodily_injuries').agg({'fraud_reported': 'sum'}).reset_index()
gpd_val3 = df2.groupby('witnesses').agg({'fraud_reported': 'sum'}).reset_index()
gpd_valc1 = df2.groupby('number_of_vehicles_involved').agg('count').reset_index()
gpd_valc2 = df2.groupby('bodily_injuries').agg('count').reset_index()
gpd_valc3 = df2.groupby('witnesses').agg('count').reset_index()
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(22, 6))
```

```
sns.barplot(x='number_of_vehicles_involved', y='fraud_reported', data=gpd_val1, ax=ax1)
sns.barplot(x='bodily_injuries', y='fraud_reported', data=gpd_val2, ax=ax3)
sns.barplot(x='witnesses', y='fraud_reported', data=gpd_val3, ax=ax2)
plt.show()
part_list = [gpd_val1, gpd_val2, gpd_val3]
counts_lst = [gpd_valc1, gpd_valc2, gpd_valc3]
srch_gp = ['number_of_vehicles_involved', 'bodily_injuries', 'witnesses']
total_list = []
for i in range(len(counts_lst)):
    temp1 = counts_lst[i]
    gby = srch_gp[i]
    temp2 = pd.concat([part_list[i], temp1['fraud_reported'].rename('Total Accidents')], axis=1)
    temp2['Percentage by {}'.format(gby)] = round((temp2['fraud_reported'] / temp2['Total Accidents']) * 100, 3)
    temp2['Percentage by Total'] = round((temp2['fraud_reported'] / sum(temp2['Total Accidents'])) * 100, 3)
    temp3 = [['Column total'],[sum(temp2['fraud_reported'])],[sum(temp2['Total Accidents'])],[sum(temp2['Percentage by {}'.format(gby)])],[sum(temp2['Percentage by Total'])]]
    nr1 = pd.DataFrame(temp3)
    nr1 = nr1.transpose()
    nr1.rename(columns={0: '{}'.format(gby), 1: 'fraud_reported', 2: 'Total Accidents', 3: 'Percentage by {}'.format(gby),4: 'Percentage by Total'}, inplace=True)
    total_list.append(pd.concat([temp2, nr1], ignore_index=True))
for ii in range(len(total_list)):
    print(tabulate(total_list[ii], headers=total_list[ii].columns))
    print(' ')
#Grafic -fraud_reported & auto_make, auto_year
```

```
gpd_val1=df2.groupby('auto_make').agg({'fraud_reported':'sum'}).reset_index()

gpd_val2=df2.groupby('auto_year').agg({'fraud_reported':'sum'}).reset_index()

fig, (ax1,ax2) = plt.subplots(1,2,figsize=(22, 6))

grph1=sns.barplot(x='auto_make', y='fraud_reported', data = gpd_val1, ax=ax1)

sns.barplot(x='auto_year', y='fraud_reported', data = gpd_val2, ax=ax2)

grph1.set_xticklabels(grph1.get_xticklabels(),rotation=45,horizontalalignment='right')

plt.show()

#Grafic -fraud_reported & insured_age

gpd_val1 = df2.groupby('insured_age').agg({'fraud_reported': 'sum'}).reset_index()

gpd_val2 = df2.groupby('insured_age').agg('count').reset_index()

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(22, 6))

sns.barplot(x='insured_age', y='fraud_reported', data=gpd_val1, ax=ax1)

sns.barplot(x='insured_age', y='fraud_reported', data=gpd_val2, ax=ax2)

plt.show()

total_list = pd.concat([gpd_val1, gpd_val2['fraud_reported'].rename('Total Accidents')], axis=1)

total_list['Percentage by insured_age'] = round((total_list['fraud_reported'] / total_list['Total Accidents']) * 100, 3)

total_list['Percentage by Total'] = round((total_list['fraud_reported'] / sum(total_list['Total Accidents'])) * 100, 3)

ax2.set(ylabel='Total counts')

data = [['Column total'],

[sum(total_list['fraud_reported'])],

[sum(total_list['Total Accidents'])],

[sum(total_list['Percentage by insured_age'])],

[sum(total_list['Percentage by Total'])]]
```

```
nr = pd.DataFrame(data)

nr1 = nr.transpose()

nr1.rename(columns={0: 'Make', 1: 'fraud_reported', 2: 'Total Accidents', 3: 'Percentage by insured_age', 4: 'Percentage by Total'},inplace=True)

pd.concat([total_list, nr1], ignore_index=True)

print(tabulate(total_list, headers=total_list.columns))

#Grafic -fraud_reported & policy_bind_year, policy_deductable, auto_year

gpd_val1=df2.groupby('policy_bind_year').agg({'fraud_reported':'sum'}).reset_index()

gpd_val2=df2.groupby('policy_deductable').agg({'fraud_reported':'sum'}).reset_index()

gpd_val3=df2.groupby('auto_year').agg({'fraud_reported':'sum'}).reset_index()

fig, (ax1,ax2,ax3) = plt.subplots(1,3,figsize=(22, 6))

sns.barplot(x='policy_bind_year', y='fraud_reported', data = gpd_val1, ax=ax1)

grph2 = sns.barplot(x='policy_deductable', y='fraud_reported', data = gpd_val2, ax=ax2)

sns.barplot(x='auto_year', y='fraud_reported', data = gpd_val3, ax=ax3)

grph2.set_xticklabels(grph2.get_xticklabels(),rotation=45,horizontalalignment='right')

plt.show()

gpd_val1=df2.groupby('policy_bind_year').agg({'fraud_reported':'sum'}).reset_index()

gpd_val2=df2.groupby('policy_deductable').agg({'fraud_reported':'sum'}).reset_index()

gpd_val3=df2.groupby('auto_year').agg({'fraud_reported':'sum'}).reset_index()

fig, (ax1,ax2,ax3) = plt.subplots(1,3,figsize=(22, 6))

sns.barplot(x='policy_bind_year', y='fraud_reported', data = gpd_val1, ax=ax1)

sns.barplot(x='policy_deductable', y='fraud_reported', data = gpd_val2, ax=ax2)

grph1 = sns.barplot(x='auto_year', y='fraud_reported', data = gpd_val3, ax=ax3)

grph1.set_xticklabels(grph1.get_xticklabels(),rotation=45,horizontalalignment='right')

plt.show()
```

```
#Grafic - fraud_reported & incident_type

gpd_val1=df2.groupby('incident_type').agg({'fraud_reported':'sum'}).reset_index()

print(gpd_val1['fraud_reported'].sum(), 'cases with fraud reported')

fig, (ax1) = plt.subplots(1,1,figsize=(22, 8))

grph1=sns.barplot(x='incident_type', y='fraud_reported', data = gpd_val1, ax=ax1)

grph1.set_xticklabels(grph1.get_xticklabels(),

rotation=45,horizontalalignment='right')

plt.show()

#Grafic - fraud_reported & incident_severity,authorities_contacted

gpd_val1=df2.groupby('incident_severity').agg({'fraud_reported':'sum'}).reset_index()

gpd_val2=df2.groupby('authorities_contacted').agg({'fraud_reported':'sum'}).reset_index()

fig, (ax1,ax2) = plt.subplots(1,2,figsize=(22, 6))

sns.barplot(x='incident_severity', y='fraud_reported', data = gpd_val1, ax=ax1)

sns.barplot(x='authorities_contacted', y='fraud_reported', data = gpd_val2, ax=ax2)

plt.show()

#Grafic - fraud_reported & insured_sex, insured_education_level, insured_age

gpd_val1=df2.groupby('insured_sex').agg({'fraud_reported':'sum'}).reset_index()

gpd_val2=df2.groupby('insured_education_level').agg({'fraud_reported':'sum'}).reset_index()

gpd_val3=df2.groupby('insured_age').agg({'fraud_reported':'sum'}).reset_index()

fig, (ax1,ax2,ax3) = plt.subplots(1,3,figsize=(22, 6))

sns.barplot(x='insured_sex', y='fraud_reported', data = gpd_val1, ax=ax1)

sns.barplot(x='insured_education_level', y='fraud_reported', data = gpd_val2, ax=ax2)

sns.barplot(x='insured_age', y='fraud_reported', data = gpd_val3, ax=ax3)

plt.show()

#Grafic - fraud_reported & authorities_contacted, bodily_injuries, auto_make
```

```
gpd_val1=df2.groupby('authorities_contacted').agg({'fraud_reported':'sum'}).reset_index()

gpd_val2=df2.groupby('bodily_injuries').agg({'fraud_reported':'sum'}).reset_index()

gpd_val3=df2.groupby('auto_make').agg({'fraud_reported':'sum'}).reset_index()

fig, (ax1,ax2,ax3) = plt.subplots(3,1,figsize=(22, 6))

sns.barplot(x='authorities_contacted', y='fraud_reported', data = gpd_val1, ax=ax1)

sns.barplot(x='bodily_injuries', y='fraud_reported', data = gpd_val2, ax=ax2)

sns.barplot(x='auto_make', y='fraud_reported', data = gpd_val3, ax=ax3)

plt.show()

#Vom elimina variabila dependenta

X=df2.drop('fraud_reported',axis=1).copy()

y=df2['fraud_reported'].copy()

#Vom codifica datelor calitative

#Pentru variabila insured_sex, unde valorile se incadreaza in intervalul {FEMALE, MALE}

cols=['insured_sex']

y_val = ['FEMALE']

x_val = ['MALE']

for i in range(len(cols)):

    X_idx1 = X[cols[i]]==y_val[i]

    X_idx2 = X[cols[i]]==x_val[i]

    X.loc[list(X_idx1),cols[i]]=1

    X.loc[list(X_idx2),cols[i]]=0

for i in range(len(cols)):

    X[cols[i]] = X[cols[i]].astype('int')

print(X.dtypes)

#Pentru variabilele calitative cu mai mult de 2 alternative de raspuns
```

```
#insured_education_level,incident_type,incident_severity,authorities_contacted,incident_state
,incident_city,auto_make

col_map = [{'insured_education_level':{'Associate':1,'College':2,'High
School':3,'JD':4,'Masters':5,'MD':6,'PhD':7}},

{'incident_type':{'Multi-vehicle Collision':1,'Parked Car':2,'Single Vehicle
Collision':3,'Vehicle Theft':4}},

{'incident_severity':{'Major Damage':1,'Minor Damage':2,'Total Loss':3,'Trivial Damage':4}},

{'authorities_contacted':{'Ambulance':1,'Fire':2,'None':0,'Other':4,'Police':5}},

{'incident_state':{'NC':1 ,'NY':2,'OH':3,'PA':4,'SC':5,'VA':6,'WV':7}},

{'incident_city':{'Arlington':1,'Columbus':2,'Hillsdale':3,'Northbend':4,'Northbrook':5,'Riverw
ood':6,'Springfield':7}},

{'auto_make':{'Accura':1,'Audi':2,'BMW':3,'Chevrolet':4,'Dodge':5,'Ford':6,'Honda':7,'Jeep':8,'
Mercedes':9,'Nissan':10,'Saab':11,'Suburu':12,'Toyota':13,'Volkswagen':14}}]

X2 = X.copy()

for i in range(len(col_map)):

    X2.replace(col_map[i], inplace=True)

#Testul Chi2

chi_result=[]

for feat in X2.columns:

    chi2, p, dof, ex = stats.chi2_contingency(pd.crosstab(X2[feat], y))

    chi_result.append([feat,chi2,p])

    ch_df = pd.DataFrame(chi_result, columns=['Feature', 'Chi2 val', 'p-val'])

    ch_df.sort_values(by=['p-val'], inplace=True)

print(tabulate(ch_df, headers=['Feature', 'Chi2 val', 'p-val']))

tempy_x2 = pd.DataFrame(X2.corrwith(y,axis=0 ).sort_values(ascending=False))

#Coeficientul de Corelatie Pearson

print(tabulate(tempy_x2, headers=['Feature', 'Pearson Correlation']))
```

```
#Impartirea in testing set si training set
X_train, X_test, y_train, y_test = train_test_split(X2, y,stratify=y, random_state=42)
clf_dt_m1 = DecisionTreeClassifier(random_state=42)
clf_dt1 = clf_dt_m1.fit(X_train, y_train)
#Arbore decizional preliminar
y_pred_gini = clf_dt1.predict(X_test)
print('Gini stats')
print("Accuracy:",metrics.accuracy_score(y_test, y_pred_gini))
print("balanced_accuracy:",metrics.balanced_accuracy_score(y_test, y_pred_gini))
print("brier_score_loss:",metrics.brier_score_loss(y_test, y_pred_gini))
print("f1_score:",metrics.f1_score(y_test,y_pred_gini))
print("recall_score:",metrics.recall_score(y_test, y_pred_gini))
print("precision_score:",metrics.precision_score(y_test, y_pred_gini))
print("roc_auc_score:",metrics.roc_auc_score(y_test, y_pred_gini))
precision, recall, thresholds = metrics.precision_recall_curve(y_test, y_pred_gini)
#Matricea de confuzie - arbore de decizie
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 7))
fig.tight_layout(pad=5.0)
metrics.plot_confusion_matrix(clf_dt1, X_test, y_test, display_labels=["Not Fraudulent Claim", "Fraudulent Claim"], ax=ax1)
tn, fp, fn, tp = metrics.confusion_matrix(y_test, y_pred_gini).ravel()
plt.show()
ax2.step(recall, precision, color='b', alpha=0.2, where='post')
ax2.fill_between(recall, precision, step='post', alpha=0.2, color='b')
ax2.set_xlabel('Recall')
```

```
ax2.set_ylabel('Precision')

ax2.set_ylim([0.0, 1.05])

ax2.set_xlim([-0.005, 1.0])

ax2.set_title('Precision-Recall curve:')

print('True Negatives:', tn)

print('False Postives:', fp)

print('False Negatives:', fn)

print('True Positive:', tp)

print('Recall:', tp/(fn+tp))

print('Precision:', tp/(fp+tp))

print('Prevalence:', (fn+tp)/(tn+fp+fn+tp))

# Compararea modelelor - Decision Trees, Random Forest, AdaBoost, and XGBoost

classifiers = {

    'DecisionTreeClassifier': DecisionTreeClassifier(random_state=42),


    'RandomForestClassifier': RandomForestClassifier(),


    'AdaBoostClassifier': AdaBoostClassifier(),


    "XGBClassifier": XGBClassifier(use_label_encoder=False,

objective='binary:logistic', eval_metric='aucpr'),

}

df_models = pd.DataFrame(columns=['model',

                                  'run_time',

                                  'avg_accy',
                                    'avg_accy_std',
                                    'avg_recall',
                                    'avg_recall_std',
                                    'avg_precision',
                                    'avg_precision_std',
                                    'avg_f1',
                                    'avg_f1_std',
                                    'avg_matthew_corcoef',
                                    'avg_matthew_corcoef_std',
                                    'avg_roc_auc',
                                    'avg_roc_auc_std',
])
scorer = {'accuracy_score': metrics.make_scorer(metrics.accuracy_score),
 'f1_score': metrics.make_scorer(metrics.f1_score),
 'recall_score': metrics.make_scorer(metrics.recall_score),
 'precision_score':
metrics.make_scorer(metrics.average_precision_score),
 'matthew_corrcoef':
metrics.make_scorer(metrics.matthews_corrcoef),
 'roc_auc_score': metrics.make_scorer(metrics.roc_auc_score)
 }
for key in classifiers:
    print('*', key)
    start_time = time.time()
    classifier = classifiers[key]
    model = classifier.fit(X_train, y_train)
    cvs = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=42)
    cv_scores = cross_validate(model, X_test, y_test, cv=cvs, scoring=scorer)
    y_pred = model.predict(X_test)
    row = {
        'model': key,
        'run_time': format(round((time.time() - start_time) / 60, 2)),
        'avg_accy': cv_scores['test_accuracy_score'].mean(),
        'avg_accy_std': cv_scores['test_accuracy_score'].std(),
        'avg_recall': cv_scores['test_recall_score'].mean(),
        'avg_recall_std': cv_scores['test_recall_score'].std(),
        'avg_precision': cv_scores['test_precision_score'].mean(),
        'avg_precision_std': cv_scores['test_precision_score'].std(),
        'avg_f1': cv_scores['test_f1_score'].mean(),
        'avg_f1_std': cv_scores['test_f1_score'].std(),
        'avg_matthew_corcoef': cv_scores['test_matthew_corrcoef'].mean(),
        'avg_matthew_corcoef_std':
cv_scores['test_matthew_corrcoef'].std(),
 'avg_roc_auc': cv_scores['test_roc_auc_score'].mean(),
 'avg_roc_auc_std': cv_scores['test_roc_auc_score'].std(),
 }
df_models = df_models.append(row, ignore_index=True)
print(df_models)
#Aplicarea SMOTE si compararea modelelor
sm = SMOTE(random_state=42)
```

```
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)

print(y_train.value_counts())

print(y_train_res.value_counts())

df_models_smote = pd.DataFrame(columns=['model',

                                    'run_time',

                                    'avg_accy',

                                    'avg_accy_std',

                                    'avg_recall',

                                    'avg_recall_std',

                                    'avg_precision',

                                    'avg_precision_std',

                                    'avg_f1',

                                    'avg_f1_std',

                                    'avg_matthew_corcoef',

                                    'avg_matthew_corcoef_std',

                                    'avg_roc_auc',

                                    'avg_roc_auc_std',

                                    ])

scorer = {'accuracy_score': metrics.make_scorer(metrics.accuracy_score),

      'f1_score': metrics.make_scorer(metrics.f1_score),

      'recall_score': metrics.make_scorer(metrics.recall_score),

      'precision_score': metrics.make_scorer(metrics.precision_score),

      'matthew_corrcoef': metrics.make_scorer(metrics.matthews_corrcoef),

      'roc_auc_score': metrics.make_scorer(metrics.roc_auc_score)

     }
```

```
  for key in classifiers:
    print('*', key)
    start_time = time.time()
    classifier = classifiers[key]
    model = classifier.fit(X_train_res, y_train_res) # <--- pass the SMOTE generate training
data set
    #scorer = metrics.make_scorer(metrics.recall_score)
    #cv_scores = cross_val_score(model, X_test, y_test, cv=5, scoring=scorer)
    cvs = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=42)
    cv_scores = cross_validate(model, X_test, y_test, cv=cvs,scoring=scorer) # <--- tested the
SMOTE trained model on original testing data
    y_pred = model.predict(X_test)
    print(model.get_params())
    row = {
      'model': key,
      'run_time': format(round((time.time() - start_time) / 60, 2)),
      'avg_accy': cv_scores['test_accuracy_score'].mean(),
      'avg_accy_std': cv_scores['test_accuracy_score'].std(),
      'avg_recall': cv_scores['test_recall_score'].mean(),
      'avg_recall_std': cv_scores['test_recall_score'].std(),
      'avg_precision': cv_scores['test_precision_score'].mean(),
      'avg_precision_std': cv_scores['test_precision_score'].std(),
      'avg_f1': cv_scores['test_f1_score'].mean(),
      'avg_f1_std': cv_scores['test_f1_score'].std(),
      'avg_matthew_corcoef': cv_scores['test_matthew_corrcoef'].mean(),
      'avg_matthew_corcoef_std': cv_scores['test_matthew_corrcoef'].std(),
        'avg_roc_auc': cv_scores['test_roc_auc_score'].mean(),

        'avg_roc_auc_std': cv_scores['test_roc_auc_score'].std(),

        }

    df_models_smote = df_models_smote.append(row, ignore_index=True)

print(df_models.head())

print(df_models_smote.head())

print(tabulate(df_models, headers=df_models.columns))

print(tabulate(df_models_smote, headers=df_models_smote.columns))
```