car-insurance-prediction

April 29, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df=pd.read_csv("D:\\projects\\DS Internship datasets\\project␣
↪2\\Car_Insurance_Claim.csv")
```

[3]:
```
df.head()
```

[3]:
```
       ID    AGE  GENDER      RACE DRIVING_EXPERIENCE    EDUCATION  \
0  569520    65+  female  majority               0-9y  high school
1  750365  16-25    male  majority               0-9y         none
2  199901  16-25  female  majority               0-9y  high school
3  478866  16-25    male  majority               0-9y   university
4  731664  26-39    male  majority             10-19y         none

         INCOME  CREDIT_SCORE  VEHICLE_OWNERSHIP VEHICLE_YEAR  MARRIED  \
0   upper class      0.629027                1.0   after 2015      0.0
1       poverty      0.357757                0.0  before 2015      0.0
2 working class      0.493146                1.0  before 2015      0.0
3 working class      0.206013                1.0  before 2015      0.0
4 working class      0.388366                1.0  before 2015      0.0

  CHILDREN  POSTAL_CODE  ANNUAL_MILEAGE VEHICLE_TYPE  SPEEDING_VIOLATIONS  \
0      1.0        10238         12000.0        sedan                    0
1      0.0        10238         16000.0        sedan                    0
2      0.0        10238         11000.0        sedan                    0
3      1.0        32765         11000.0        sedan                    0
4      0.0        32765         12000.0        sedan                    2

  DUIS  PAST_ACCIDENTS  OUTCOME
0    0               0      0.0
1    0               0      1.0
2    0               0      0.0
```

```
3      0                0       0.0
4      0                1       1.0
```

[4]:
```
df.tail()
```

[4]:
```
          ID    AGE  GENDER      RACE DRIVING_EXPERIENCE    EDUCATION  \
9995  323164  26-39  female  majority             10-19y   university
9996  910346  26-39  female  majority             10-19y         none
9997  468409  26-39    male  majority               0-9y  high school
9998  903459  26-39  female  majority             10-19y  high school
9999  442696  26-39  female  majority               0-9y         none

             INCOME  CREDIT_SCORE  VEHICLE_OWNERSHIP VEHICLE_YEAR  MARRIED  \
9995    upper class      0.582787                1.0  before 2015      0.0
9996   middle class      0.522231                1.0   after 2015      0.0
9997   middle class      0.470940                1.0  before 2015      0.0
9998        poverty      0.364185                0.0  before 2015      0.0
9999  working class      0.435225                1.0  before 2015      1.0

      CHILDREN  POSTAL_CODE  ANNUAL_MILEAGE VEHICLE_TYPE  SPEEDING_VIOLATIONS  \
9995       0.0        10238         16000.0        sedan                    0
9996       1.0        32765             NaN        sedan                    1
9997       1.0        10238         14000.0        sedan                    0
9998       1.0        10238         13000.0        sedan                    2
9999       1.0        10238         13000.0        sedan                    0

      DUIS  PAST_ACCIDENTS  OUTCOME
9995     0               1      0.0
9996     0               0      0.0
9997     0               0      0.0
9998     0               1      1.0
9999     0               0      0.0
```

[5]:
```
df.describe()
```

[5]:
```
                  ID  CREDIT_SCORE  VEHICLE_OWNERSHIP       MARRIED  \
count   10000.000000   9018.000000       10000.000000  10000.000000
mean   500521.906800      0.515813           0.697000      0.498200
std    290030.768758      0.137688           0.459578      0.500022
min       101.000000      0.053358           0.000000      0.000000
25%    249638.500000      0.417191           0.000000      0.000000
50%    501777.000000      0.525033           1.000000      0.000000
75%    753974.500000      0.618312           1.000000      1.000000
max    999976.000000      0.960819           1.000000      1.000000

           CHILDREN   POSTAL_CODE  ANNUAL_MILEAGE  SPEEDING_VIOLATIONS  \
count  10000.000000  10000.000000     9043.000000         10000.000000
```

```
mean       0.688800  19864.548400    11697.003207             1.482900
std        0.463008  18915.613855     2818.434528             2.241966
min        0.000000  10238.000000     2000.000000             0.000000
25%        0.000000  10238.000000    10000.000000             0.000000
50%        1.000000  10238.000000    12000.000000             0.000000
75%        1.000000  32765.000000    14000.000000             2.000000
max        1.000000  92101.000000    22000.000000            22.000000

              DUIS  PAST_ACCIDENTS       OUTCOME
count  10000.00000    10000.000000  10000.000000
mean       0.23920        1.056300      0.313300
std        0.55499        1.652454      0.463858
min        0.00000        0.000000      0.000000
25%        0.00000        0.000000      0.000000
50%        0.00000        0.000000      0.000000
75%        0.00000        2.000000      1.000000
max        6.00000       15.000000      1.000000
```

[6]:
```
df.isnull().any().sum()
```

[6]:
```
2
```

[7]:
```
df.isnull().sum()
```

[7]:
```
ID                       0
AGE                      0
GENDER                   0
RACE                     0
DRIVING_EXPERIENCE       0
EDUCATION                0
INCOME                   0
CREDIT_SCORE           982
VEHICLE_OWNERSHIP        0
VEHICLE_YEAR             0
MARRIED                  0
CHILDREN                 0
POSTAL_CODE              0
ANNUAL_MILEAGE         957
VEHICLE_TYPE             0
SPEEDING_VIOLATIONS      0
DUIS                     0
PAST_ACCIDENTS           0
OUTCOME                  0
dtype: int64
```

[8]:
```
df.shape
```

[8]: (10000, 19)

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 19 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   10000 non-null  int64
 1   AGE                  10000 non-null  object
 2   GENDER               10000 non-null  object
 3   RACE                 10000 non-null  object
 4   DRIVING_EXPERIENCE   10000 non-null  object
 5   EDUCATION            10000 non-null  object
 6   INCOME               10000 non-null  object
 7   CREDIT_SCORE         9018 non-null   float64
 8   VEHICLE_OWNERSHIP    10000 non-null  float64
 9   VEHICLE_YEAR         10000 non-null  object
 10  MARRIED              10000 non-null  float64
 11  CHILDREN             10000 non-null  float64
 12  POSTAL_CODE          10000 non-null  int64
 13  ANNUAL_MILEAGE       9043 non-null   float64
 14  VEHICLE_TYPE         10000 non-null  object
 15  SPEEDING_VIOLATIONS  10000 non-null  int64
 16  DUIS                 10000 non-null  int64
 17  PAST_ACCIDENTS       10000 non-null  int64
 18  OUTCOME              10000 non-null  float64
dtypes: float64(6), int64(5), object(8)
memory usage: 1.4+ MB
```

[10]:
```
df[df.isnull().any(axis=1)]
```

[10]:
```
          ID    AGE  GENDER      RACE DRIVING_EXPERIENCE    EDUCATION  \
13    569640  16-25  female  majority               0-9y   university
15    906223  26-39  female  majority               0-9y  high school
16    517747    65+    male  majority               30y+   university
17     24851  16-25    male  majority               0-9y         none
18    104086  26-39  female  majority               0-9y   university
…        …      …       …         …                  …            …
9977  794068    65+    male  minority               0-9y         none
9981  366048  26-39    male  majority               0-9y  high school
9985  595418  16-25    male  minority               0-9y  high school
9988  479789  26-39    male  majority             10-19y  high school
9996  910346  26-39  female  majority             10-19y         none

             INCOME  CREDIT_SCORE  VEHICLE_OWNERSHIP VEHICLE_YEAR  MARRIED  \
```

```
13      upper class       0.591260                1.0  before 2015      0.0
15      upper class       0.762798                0.0   after 2015      1.0
16      upper class       0.796175                1.0  before 2015      1.0
17          poverty            NaN                0.0  before 2015      1.0
18      upper class       0.680594                1.0  before 2015      0.0
…             …            …                …          …      …
9977    upper class       0.710640                1.0   after 2015      0.0
9981  working class            NaN                1.0  before 2015      0.0
9985  working class            NaN                1.0  before 2015      0.0
9988        poverty            NaN                0.0  before 2015      0.0
9996   middle class       0.522231                1.0   after 2015      0.0

      CHILDREN  POSTAL_CODE  ANNUAL_MILEAGE VEHICLE_TYPE  SPEEDING_VIOLATIONS  \
13         1.0        10238             NaN        sedan                    0
15         0.0        10238             NaN        sedan                    0
16         1.0        32765             NaN        sedan                   10
17         0.0        32765         12000.0        sedan                    0
18         1.0        32765             NaN        sedan                    0
…          …          …             …          …                  …
9977       1.0        32765             NaN        sedan                    0
9981       1.0        10238         11000.0        sedan                    0
9985       1.0        10238         11000.0        sedan                    0
9988       0.0        10238             NaN        sedan                    1
9996       1.0        32765             NaN        sedan                    1

      DUIS  PAST_ACCIDENTS  OUTCOME
13       0               0      0.0
15       0               0      0.0
16       2               1      0.0
17       0               0      1.0
18       0               0      1.0
…     …             …       …
9977     0               0      0.0
9981     0               0      0.0
9985     0               0      0.0
9988     0               2      1.0
9996     0               0      0.0

[1851 rows x 19 columns]
```

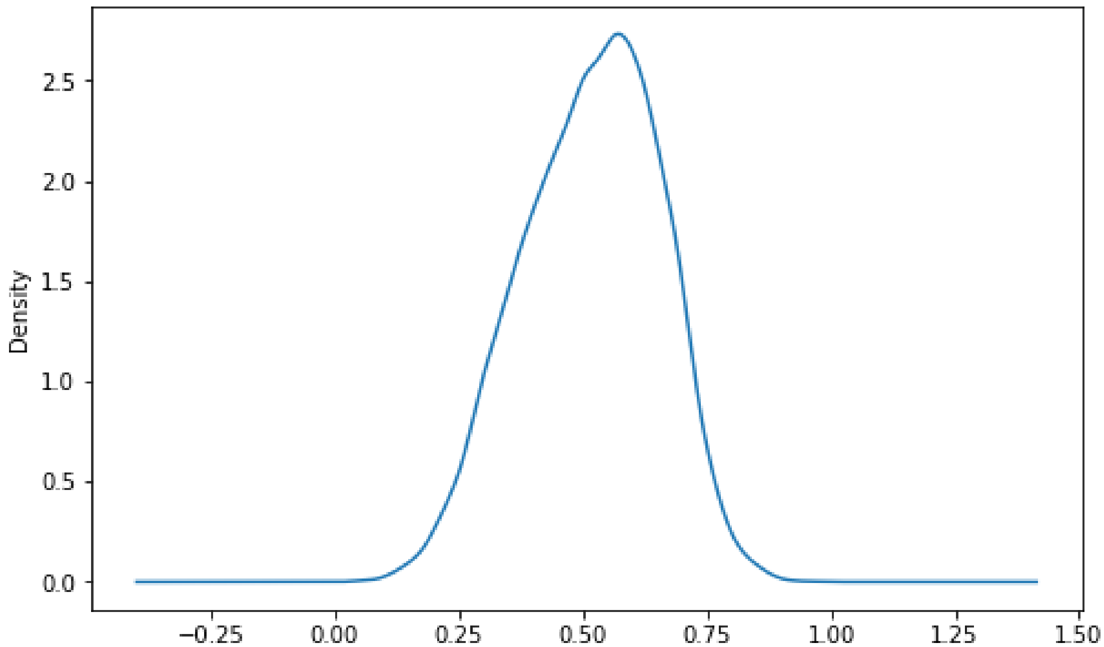
[11]:
```
df[df["CREDIT_SCORE"].isnull()]
```

[11]:
```
          ID    AGE  GENDER      RACE DRIVING_EXPERIENCE    EDUCATION  \
17     24851  16-25    male  majority               0-9y         none
23       217  16-25    male  majority               0-9y         none
37    511757  40-64  female  majority             10-19y         none
38    429947    65+    male  majority               30y+   university
```

```
47    921097  40-64  female  majority           20-29y   university
…       …     …      …        …                   …          …
9952  870405  40-64  female  majority           10-19y   university
9967   27406  26-39  female  majority           10-19y  high school
9981  366048  26-39    male  majority             0-9y  high school
9985  595418  16-25    male  minority             0-9y  high school
9988  479789  26-39    male  majority           10-19y  high school

             INCOME  CREDIT_SCORE  VEHICLE_OWNERSHIP VEHICLE_YEAR  MARRIED  \
17          poverty           NaN                0.0  before 2015      1.0
23          poverty           NaN                0.0  before 2015      0.0
37     middle class           NaN                1.0  before 2015      1.0
38      upper class           NaN                0.0   after 2015      0.0
47      upper class           NaN                1.0   after 2015      1.0
…               …             …                  …            …        …
9952    upper class           NaN                1.0   after 2015      1.0
9967   middle class           NaN                0.0  before 2015      0.0
9981  working class           NaN                1.0  before 2015      0.0
9985  working class           NaN                1.0  before 2015      0.0
9988        poverty           NaN                0.0  before 2015      0.0

      CHILDREN  POSTAL_CODE  ANNUAL_MILEAGE VEHICLE_TYPE  SPEEDING_VIOLATIONS  \
17         0.0        32765         12000.0        sedan                    0
23         0.0        10238         17000.0        sedan                    0
37         1.0        10238         11000.0        sedan                    2
38         1.0        10238         12000.0   sports car                    6
47         1.0        92101         11000.0        sedan                    3
…          …            …               …            …                    …
9952       1.0        32765          5000.0        sedan                    1
9967       0.0        92101         13000.0        sedan                    1
9981       1.0        10238         11000.0        sedan                    0
9985       1.0        10238         11000.0        sedan                    0
9988       0.0        10238             NaN        sedan                    1

      DUIS  PAST_ACCIDENTS  OUTCOME
17       0               0      1.0
23       0               0      0.0
37       0               1      0.0
38       0               5      0.0
47       0               2      0.0
…     …                …        …
9952     0               0      0.0
9967     0               0      0.0
9981     0               0      0.0
9985     0               0      0.0
9988     0               2      1.0
```

[982 rows x 19 columns]

[12]:
```
plt.figure(figsize=(8,5))
df['CREDIT_SCORE'].plot(kind='kde')
plt.show()
```



[13]:
```
df["CREDIT_SCORE"].fillna(df["CREDIT_SCORE"].median(),inplace=True)
```

[14]:
```
df["ANNUAL_MILEAGE"].fillna(df["ANNUAL_MILEAGE"].median(),inplace=True)
```

[15]:
```
df.isna().sum()
```

[15]:
```
ID                     0
AGE                    0
GENDER                 0
RACE                   0
DRIVING_EXPERIENCE     0
EDUCATION              0
INCOME                 0
CREDIT_SCORE           0
VEHICLE_OWNERSHIP      0
VEHICLE_YEAR           0
MARRIED                0
CHILDREN               0
POSTAL_CODE            0
```

```
ANNUAL_MILEAGE         0
VEHICLE_TYPE           0
SPEEDING_VIOLATIONS    0
DUIS                   0
PAST_ACCIDENTS         0
OUTCOME                0
dtype: int64
```

[16]:
```python
df.drop(["ID"],axis=1,inplace=True)
```

[17]:
```python
df.head()
```

[17]:
```
     AGE  GENDER      RACE DRIVING_EXPERIENCE    EDUCATION         INCOME  \
0    65+  female  majority               0-9y  high school    upper class
1  16-25    male  majority               0-9y         none        poverty
2  16-25  female  majority               0-9y  high school  working class
3  16-25    male  majority               0-9y   university  working class
4  26-39    male  majority             10-19y         none  working class

   CREDIT_SCORE  VEHICLE_OWNERSHIP VEHICLE_YEAR  MARRIED  CHILDREN  \
0      0.629027                1.0   after 2015      0.0       1.0
1      0.357757                0.0  before 2015      0.0       0.0
2      0.493146                1.0  before 2015      0.0       0.0
3      0.206013                1.0  before 2015      0.0       1.0
4      0.388366                1.0  before 2015      0.0       0.0

   POSTAL_CODE  ANNUAL_MILEAGE VEHICLE_TYPE  SPEEDING_VIOLATIONS  DUIS  \
0        10238         12000.0        sedan                    0     0
1        10238         16000.0        sedan                    0     0
2        10238         11000.0        sedan                    0     0
3        32765         11000.0        sedan                    0     0
4        32765         12000.0        sedan                    2     0

   PAST_ACCIDENTS  OUTCOME
0               0      0.0
1               0      1.0
2               0      0.0
3               0      0.0
4               1      1.0
```

[18]:
```python
df["AGE"].unique()#discrete
```

[18]:
```
array(['65+', '16-25', '26-39', '40-64'], dtype=object)
```

[19]:
```python
df["GENDER"].unique()#nominal
```

[19]:
```
array(['female', 'male'], dtype=object)
```

[20]:
```
df["RACE"].unique()#nominal
```

[20]:
```
array(['majority', 'minority'], dtype=object)
```

[21]:
```
df["DRIVING_EXPERIENCE"].unique()#discrete
```

[21]:
```
array(['0-9y', '10-19y', '20-29y', '30y+'], dtype=object)
```

[22]:
```
df["EDUCATION"].unique()#ordinal
```

[22]:
```
array(['high school', 'none', 'university'], dtype=object)
```

[23]:
```
df["INCOME"].unique()#ordinal
```

[23]:
```
array(['upper class', 'poverty', 'working class', 'middle class'],
      dtype=object)
```

[24]:
```
df["CREDIT_SCORE"].unique()#continous
```

[24]:
```
array([0.62902731, 0.35775712, 0.49314579, …, 0.47094023, 0.36418478,
       0.43522478])
```

[25]:
```
df["VEHICLE_OWNERSHIP"].unique()#discrete
```

[25]:
```
array([1., 0.])
```

[26]:
```
df["VEHICLE_YEAR"].unique()#nominal
```

[26]:
```
array(['after 2015', 'before 2015'], dtype=object)
```

[27]:
```
df["MARRIED"].unique()#discrete
```

[27]:
```
array([0., 1.])
```

[28]:
```
df["CHILDREN"].unique()#discrete
```

[28]:
```
array([1., 0.])
```

[29]:
```
df["POSTAL_CODE"].unique()#nominal
```

[29]:
```
array([10238, 32765, 92101, 21217], dtype=int64)
```

[30]:
```
df["ANNUAL_MILEAGE"].unique()#discrete
```

[30]:
```
array([12000., 16000., 11000., 13000., 14000., 10000.,  8000., 18000.,
       17000.,  7000., 15000.,  9000.,  5000.,  6000., 19000.,  4000.,
        3000.,  2000., 20000., 21000., 22000.])
```

[31]:
```
df["VEHICLE_TYPE"].unique()#nominal
```

[31]:
```
array(['sedan', 'sports car'], dtype=object)
```

[32]:
```
df["SPEEDING_VIOLATIONS"].unique()#continous
```

[32]:
```
array([ 0,  2,  3,  7,  6,  4, 10, 13,  1,  5,  9,  8, 12, 11, 15, 17, 19,
       18, 16, 14, 22], dtype=int64)
```

[33]:
```
df["DUIS"].unique()#continous
```

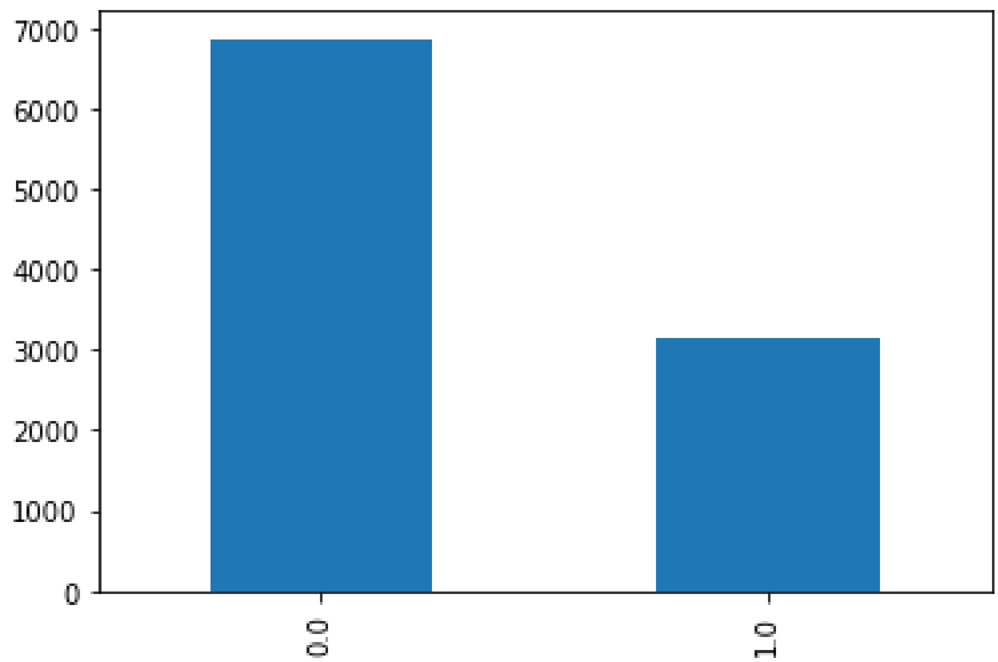
[33]:
```
array([0, 2, 1, 3, 4, 5, 6], dtype=int64)
```

[34]:
```
df["PAST_ACCIDENTS"].unique()#continous
```

[34]:
```
array([ 0,  1,  3,  7,  2,  5,  4,  6,  8, 10, 11,  9, 12, 14, 15],
      dtype=int64)
```

[35]:
```
df["OUTCOME"].value_counts().plot(kind='bar')#discrete
```

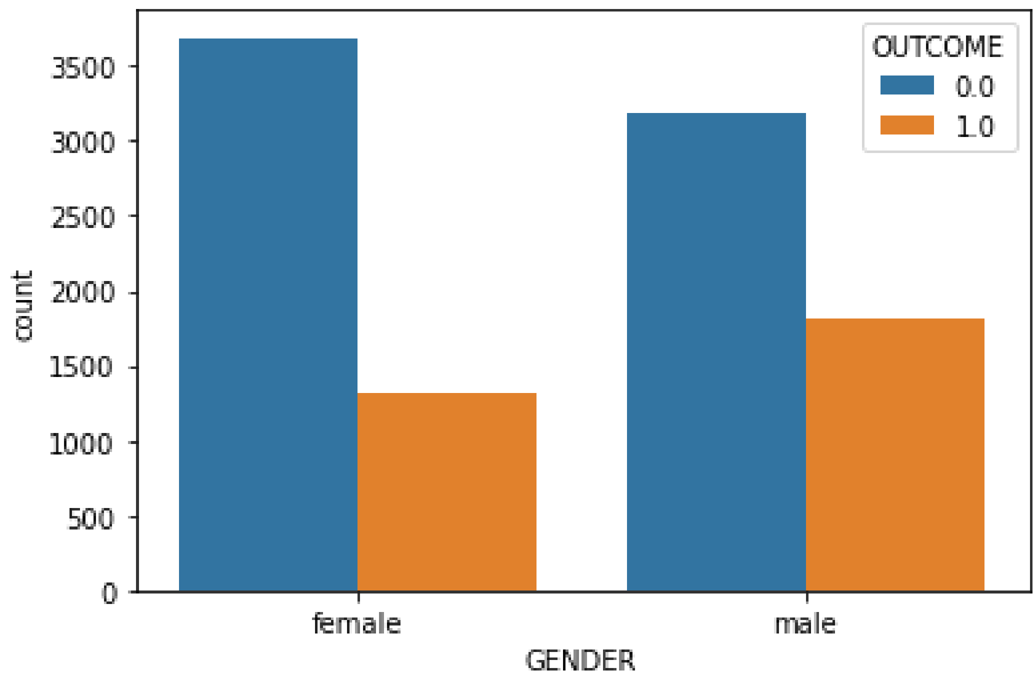
[35]:
```
<AxesSubplot:>
```



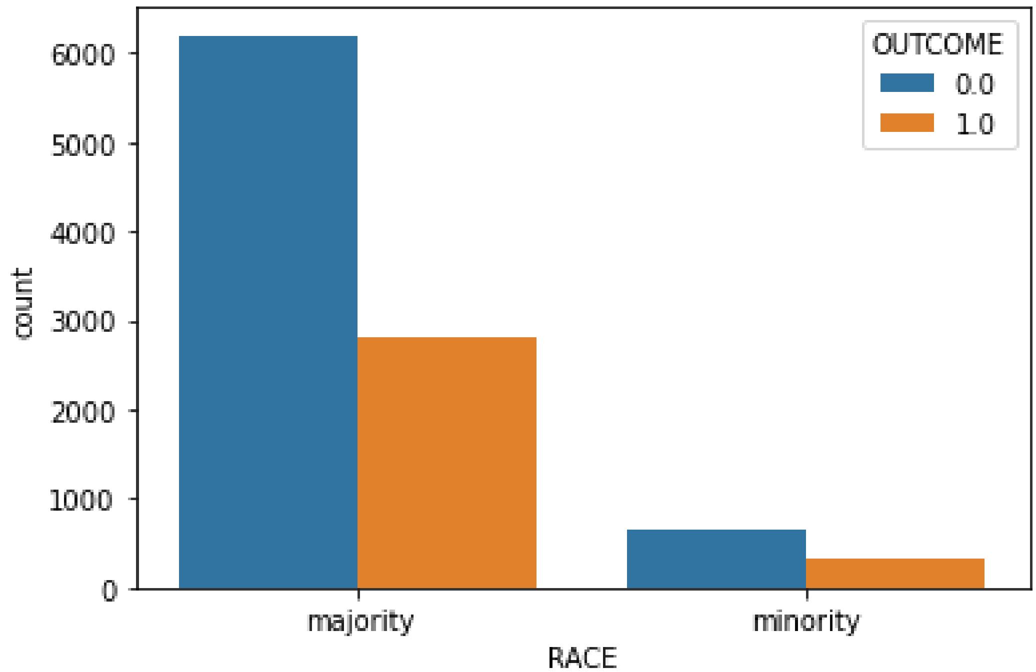
[36]:
```
df["GENDER"].value_counts()
```

[36]:
```
female    5010
male      4990
Name: GENDER, dtype: int64
```

[37]:
```
sns.countplot(data=df,x="GENDER",hue="OUTCOME")
plt.show()
```



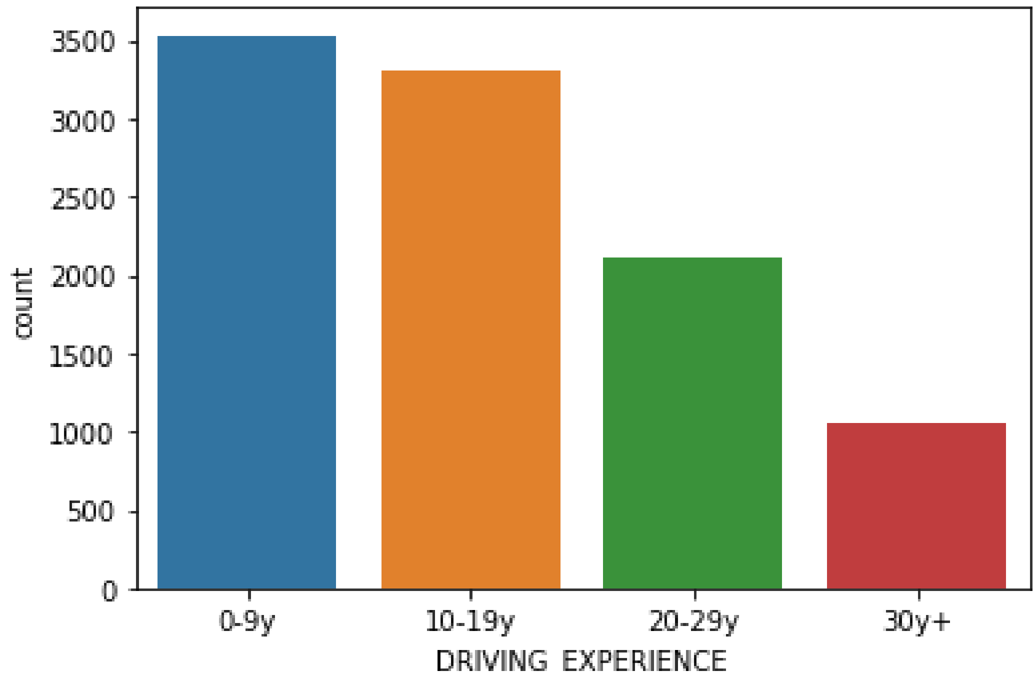
[38]:
```
sns.countplot(data=df,x="RACE",hue="OUTCOME")
plt.show()
```



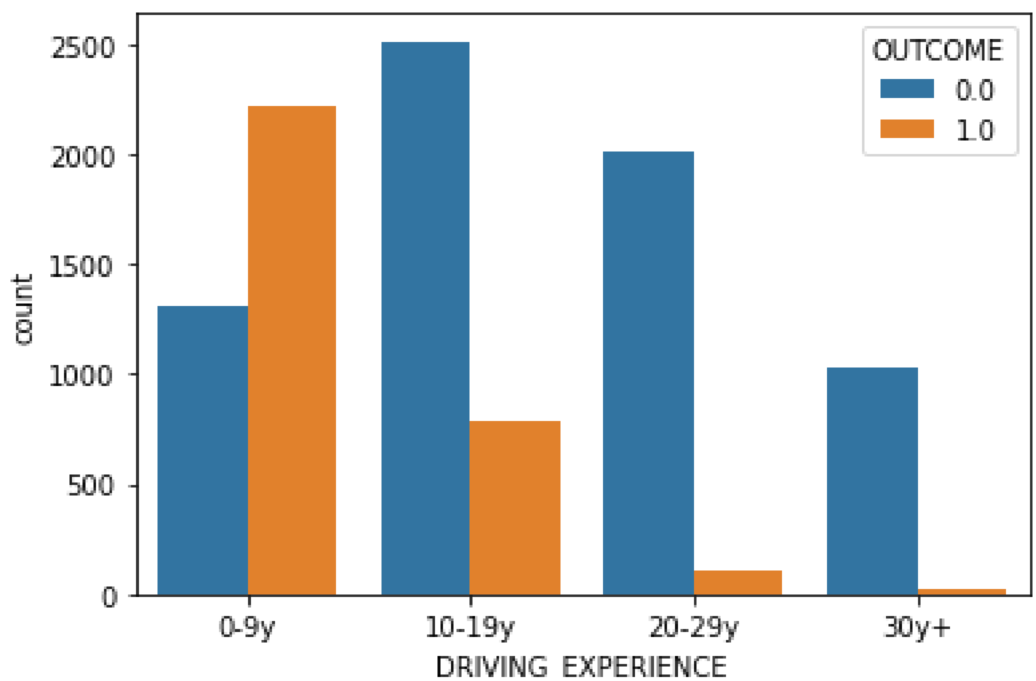
[39]:
```
df["DRIVING_EXPERIENCE"].value_counts()
```

[39]:
```
0-9y      3530
10-19y    3299
20-29y    2119
30y+      1052
Name: DRIVING_EXPERIENCE, dtype: int64
```

[40]:
```
sns.countplot(df["DRIVING_EXPERIENCE"])
plt.show()
```



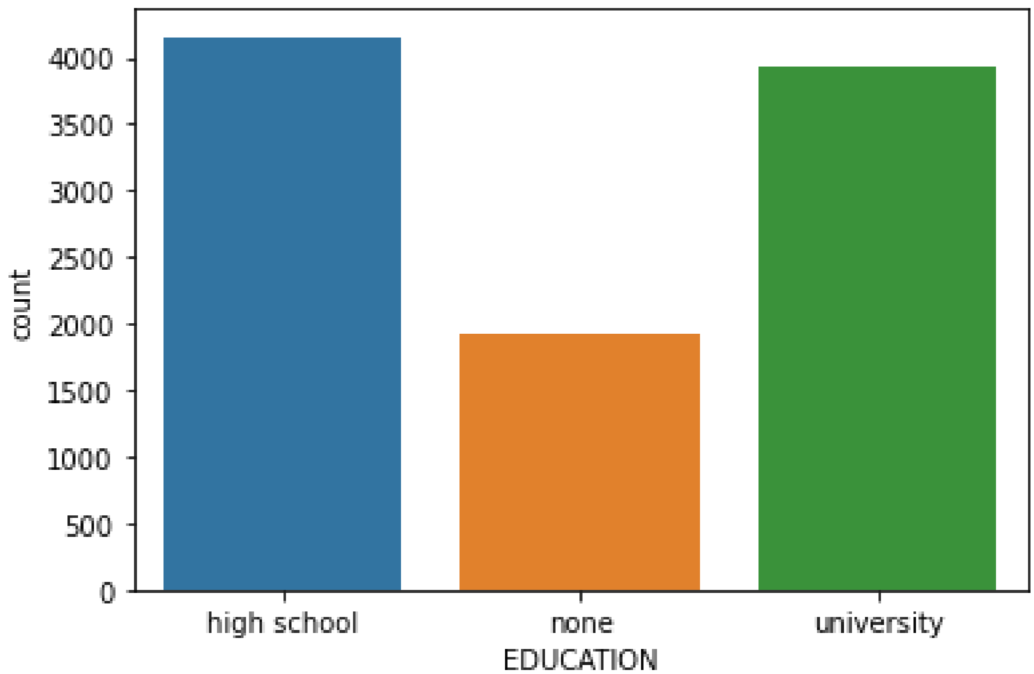
[41]:
```
sns.countplot(data=df,x="DRIVING_EXPERIENCE",hue="OUTCOME")
plt.show()
```



[42]:
```
df["EDUCATION"].value_counts()
```

[42]:
```
high school    4157
university     3928
none           1915
Name: EDUCATION, dtype: int64
```

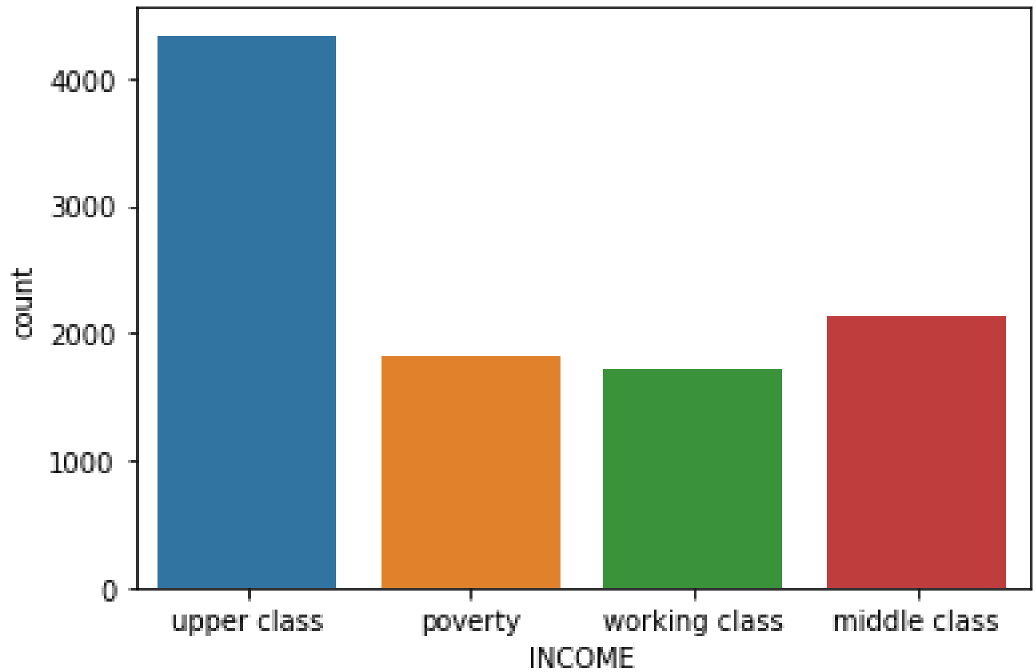
[43]:
```
sns.countplot(df["EDUCATION"])
plt.show()
```



[44]:
```
df["INCOME"].value_counts()
```

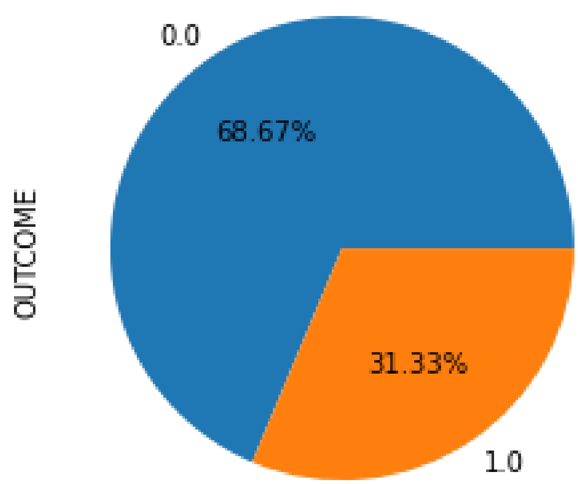
[44]:
```
upper class      4336
middle class     2138
poverty          1814
working class    1712
Name: INCOME, dtype: int64
```

[45]:
```
sns.countplot(df["INCOME"])
plt.show()
```



[46]:
```
df["OUTCOME"].value_counts().plot(kind='pie',autopct="%0.2f%%")
```

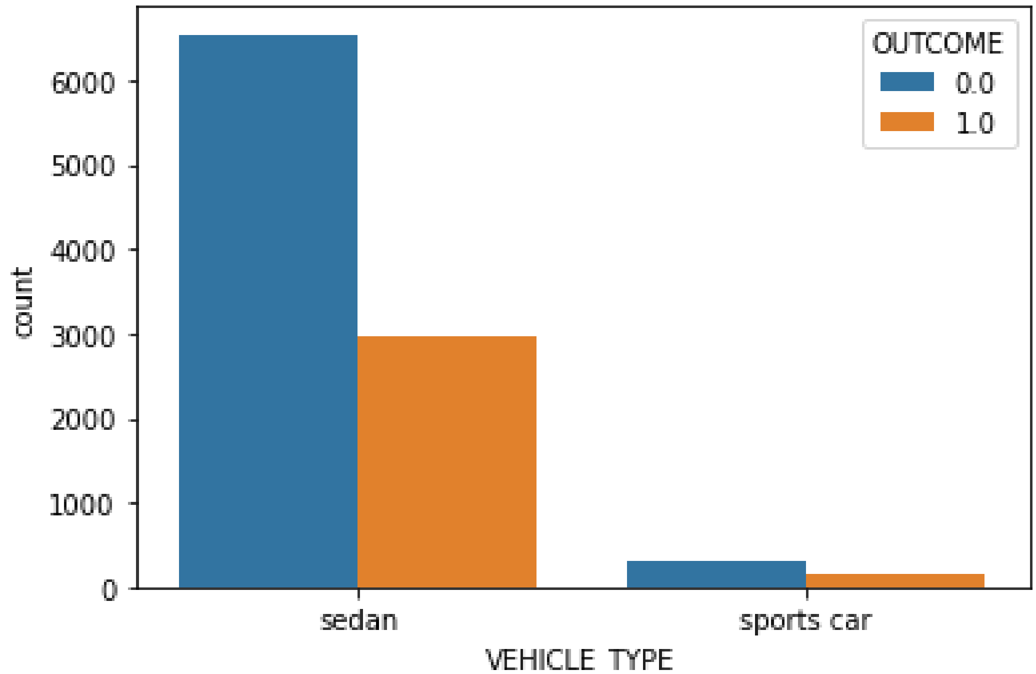
[46]: <AxesSubplot:ylabel='OUTCOME'>



[47]:
```
df["VEHICLE_TYPE"].value_counts()
```

[47]:
```
sedan         9523
sports car     477
Name: VEHICLE_TYPE, dtype: int64
```

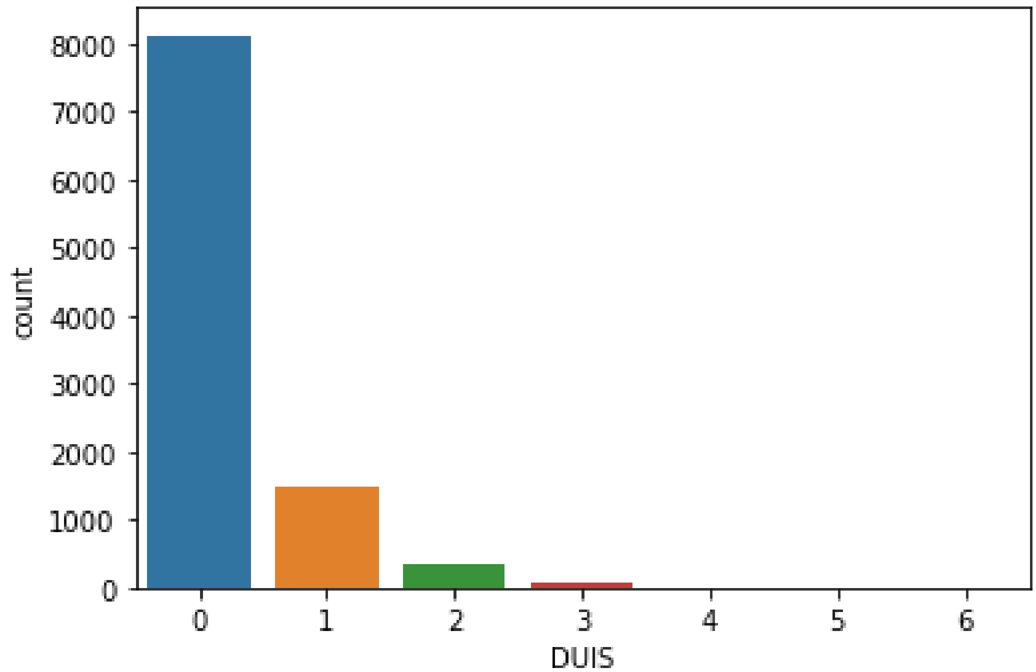
[48]:
```
sns.countplot(data=df,x='VEHICLE_TYPE',hue='OUTCOME')
plt.show()
```



[49]:
```
df["DUIS"].value_counts()
```

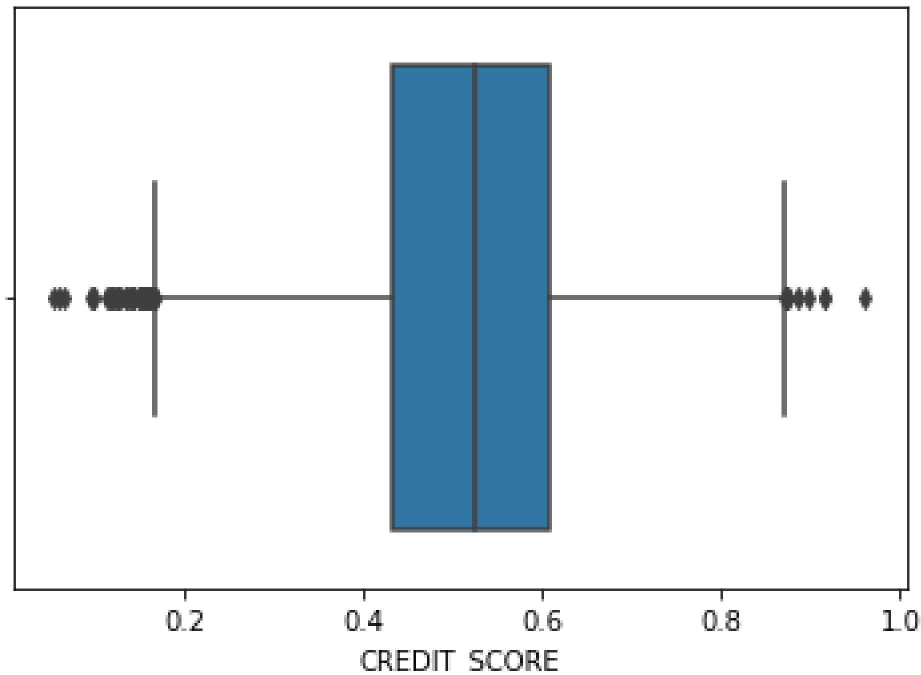
[49]:
```
0    8118
1    1470
2     331
3      68
4      10
5       2
6       1
Name: DUIS, dtype: int64
```

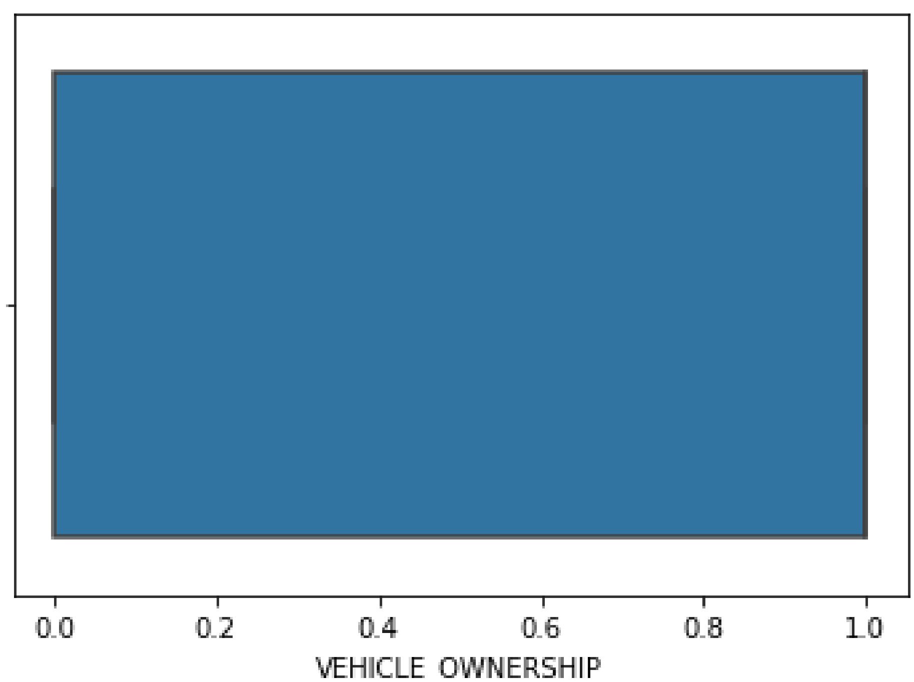
[50]:
```
sns.countplot(df["DUIS"])
plt.show()
```

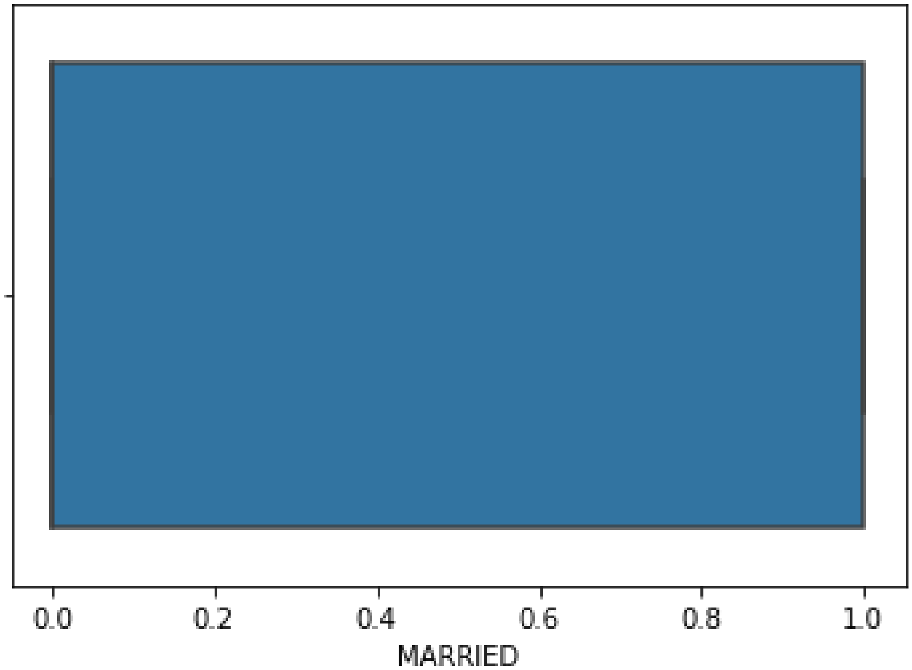


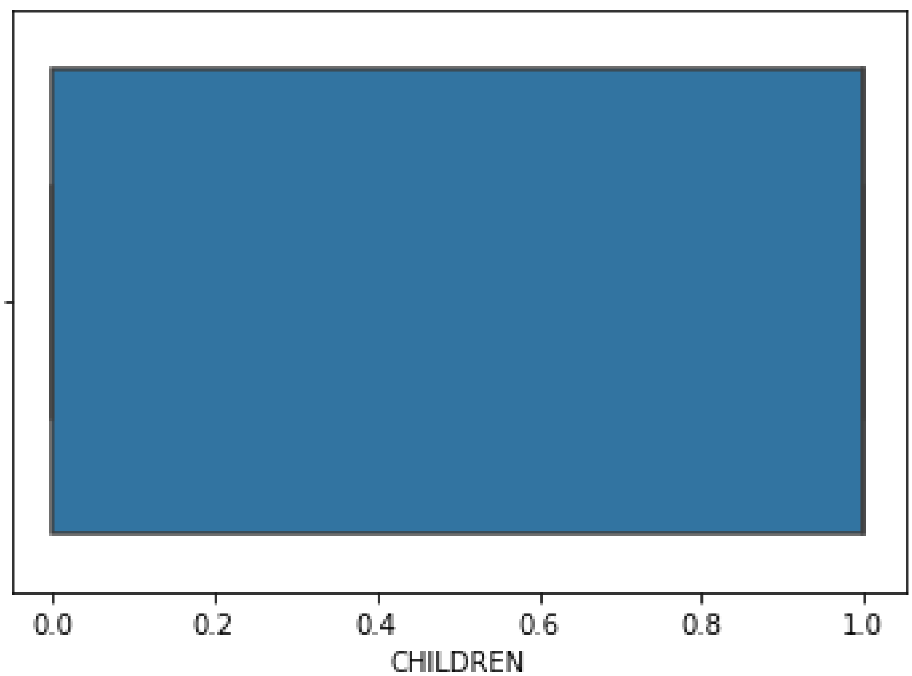
[51]:
```
n=df.select_dtypes(exclude='object')
```

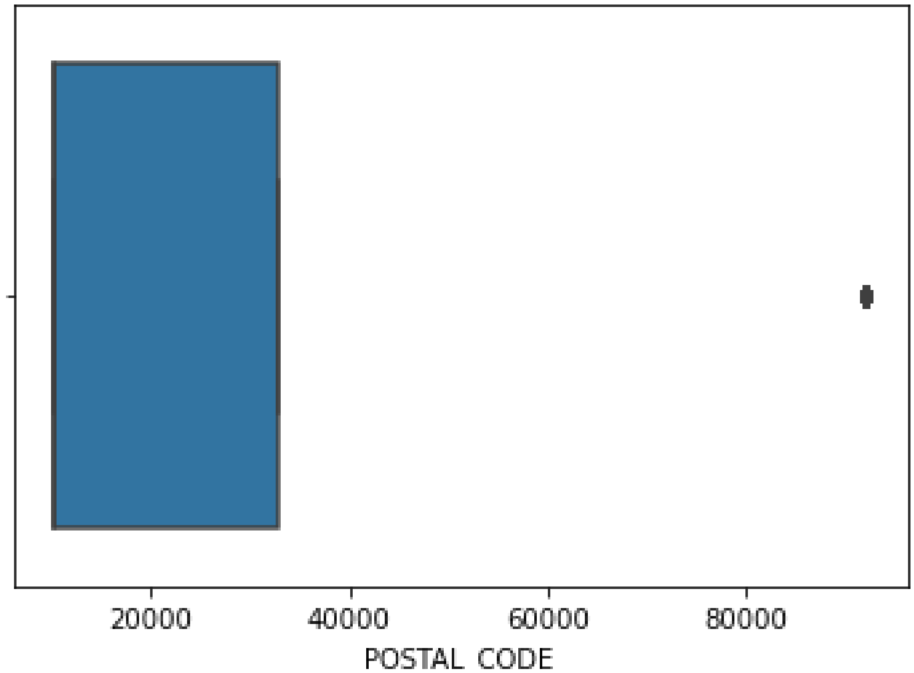
[52]:
```
for i in n.columns:
    sns.boxplot(data=n,x=i)
    plt.show()
```

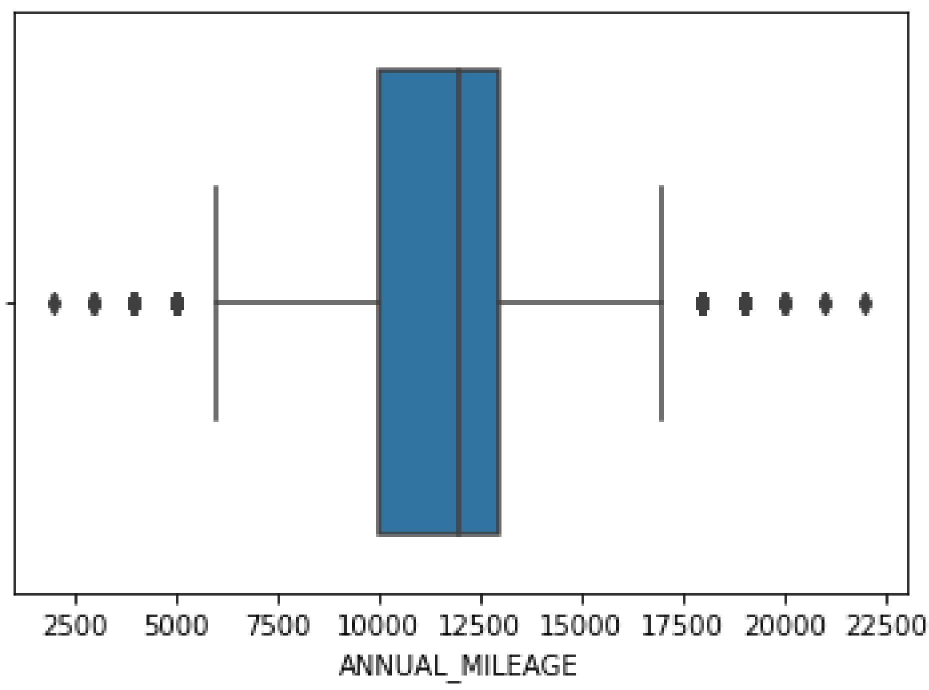

0.2
0.4
0.6
0.8
1.0
CREDIT_SCORE

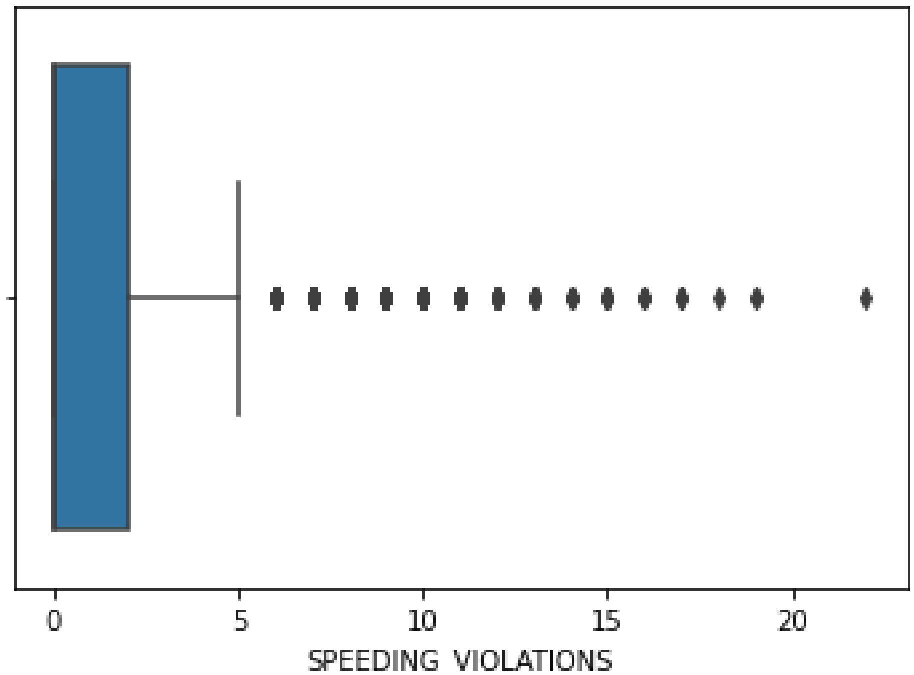

0.0
0.2
0.4
0.6
0.8
1.0
VEHICLE_OWNERSHIP

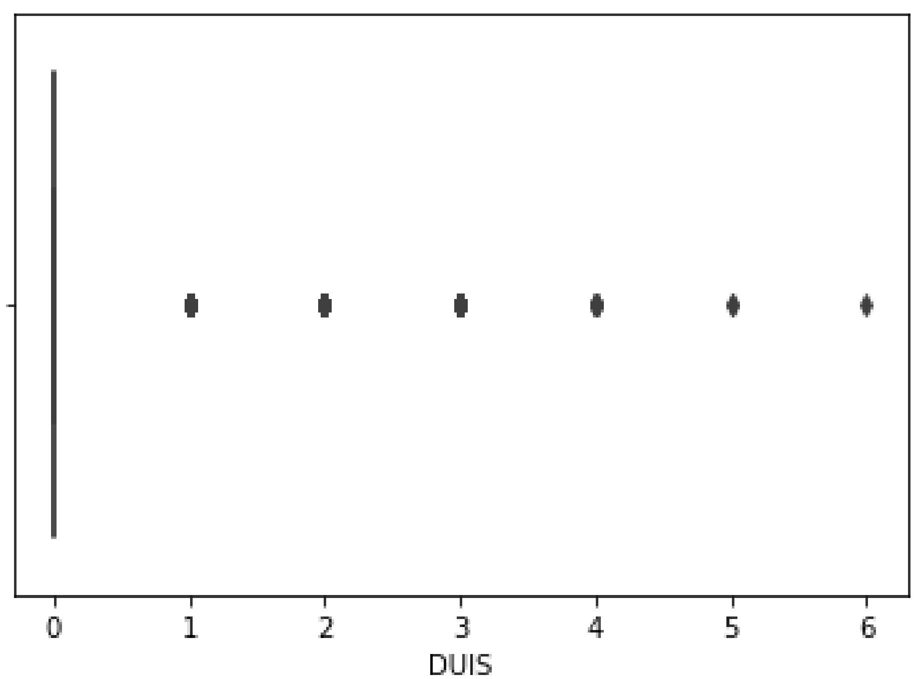

0.0
0.2
0.4
0.6
0.8
1.0
MARRIED

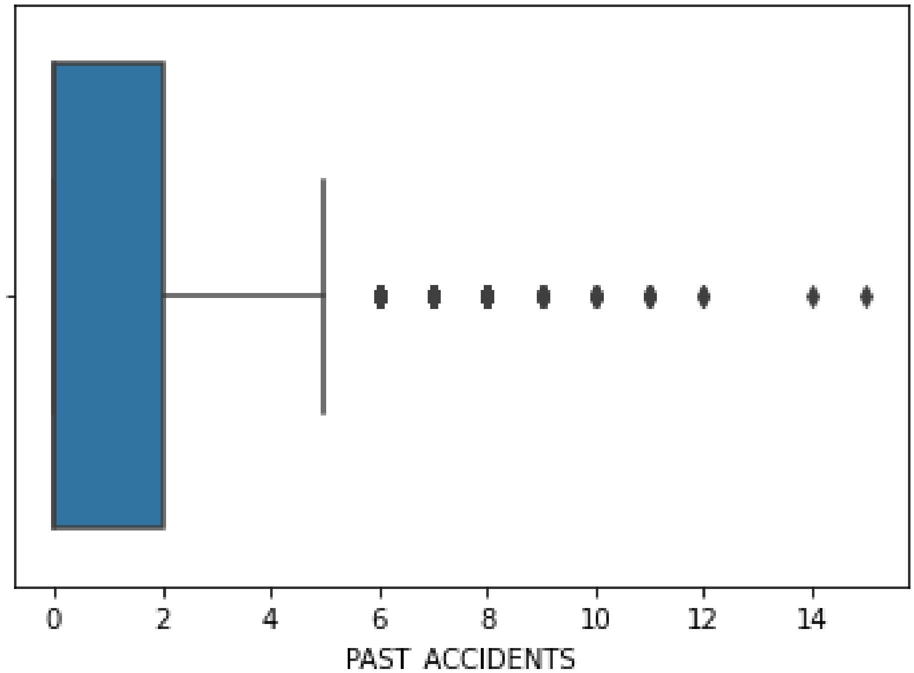

0.0
0.2
0.4
0.6
0.8
1.0
CHILDREN

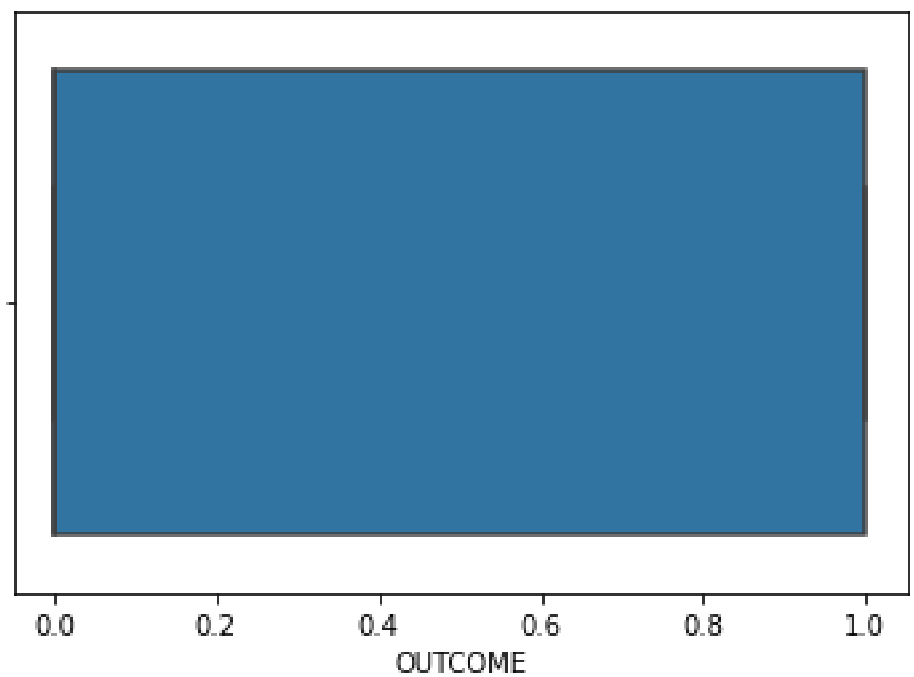

20000
40000
60000
80000
POSTAL_CODE


2500
5000
7500
10000
12500
15000
17500
20000
22500
ANNUAL_MILEAGE


0
5
10
15
20
SPEEDING_VIOLATIONS


0
1
2
3
4
5
6
DUIS





[53]:
```
df.dtypes
```

[53]:
```
AGE                      object
GENDER                   object
RACE                     object
DRIVING_EXPERIENCE       object
EDUCATION                object
INCOME                   object
CREDIT_SCORE            float64
VEHICLE_OWNERSHIP       float64
VEHICLE_YEAR             object
MARRIED                 float64
CHILDREN                float64
POSTAL_CODE               int64
ANNUAL_MILEAGE          float64
VEHICLE_TYPE             object
SPEEDING_VIOLATIONS       int64
DUIS                      int64
PAST_ACCIDENTS            int64
OUTCOME                 float64
dtype: object
```

[54]:
```
from sklearn.preprocessing import LabelEncoder
lr=LabelEncoder()
```

[55]:
```
df['AGE']=lr.fit_transform(df['AGE'])
df['GENDER']=lr.fit_transform(df['GENDER'])
df['RACE']=lr.fit_transform(df['RACE'])
df['DRIVING_EXPERIENCE']=lr.fit_transform(df['DRIVING_EXPERIENCE'])
df['EDUCATION']=lr.fit_transform(df['EDUCATION'])
df['INCOME']=lr.fit_transform(df['INCOME'])
df['VEHICLE_YEAR']=lr.fit_transform(df['VEHICLE_YEAR'])
df['VEHICLE_TYPE']=lr.fit_transform(df['VEHICLE_TYPE'])
```

[56]:
```
df.dtypes
```

[56]:
```
AGE                       int32
GENDER                    int32
RACE                      int32
DRIVING_EXPERIENCE        int32
EDUCATION                 int32
INCOME                    int32
CREDIT_SCORE            float64
VEHICLE_OWNERSHIP       float64
VEHICLE_YEAR              int32
MARRIED                 float64
CHILDREN                float64
POSTAL_CODE               int64
ANNUAL_MILEAGE          float64
```

```
VEHICLE_TYPE             int32
SPEEDING_VIOLATIONS      int64
DUIS                     int64
PAST_ACCIDENTS           int64
OUTCOME                float64
dtype: object
```

[57]:
```
df.head()
```

[57]:
```
   AGE  GENDER  RACE  DRIVING_EXPERIENCE  EDUCATION  INCOME  CREDIT_SCORE  \
0    3       0     0                   0          0       2      0.629027
1    0       1     0                   0          1       1      0.357757
2    0       0     0                   0          0       3      0.493146
3    0       1     0                   0          2       3      0.206013
4    1       1     0                   1          1       3      0.388366

   VEHICLE_OWNERSHIP  VEHICLE_YEAR  MARRIED  CHILDREN  POSTAL_CODE  \
0                1.0             0      0.0       1.0        10238
1                0.0             1      0.0       0.0        10238
2                1.0             1      0.0       0.0        10238
3                1.0             1      0.0       1.0        32765
4                1.0             1      0.0       0.0        32765

   ANNUAL_MILEAGE  VEHICLE_TYPE  SPEEDING_VIOLATIONS  DUIS  PAST_ACCIDENTS  \
0         12000.0             0                    0     0               0
1         16000.0             0                    0     0               0
2         11000.0             0                    0     0               0
3         11000.0             0                    0     0               0
4         12000.0             0                    2     0               1

   OUTCOME
0      0.0
1      1.0
2      0.0
3      0.0
4      1.0
```

[58]:
```
car_insurance=df.values
```

[59]:
```
df.shape
```

[59]: (10000, 18)

[60]:
```
x=car_insurance[:,0:17]
y=car_insurance[:,17]
```

[61]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

1 Logistic Regression

[62]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import␣
↪classification_report,confusion_matrix,accuracy_score
```

[63]:
```
logistic_model = LogisticRegression().fit(x_train,y_train)
ypredicted = logistic_model.predict(x_test)
ypredicted
```

[63]:
```
array([1., 0., 0., …, 0., 0., 0.])
```

2 Evaluation for Logistic Regression

[64]:
```
Varience=np.var(ypredicted)
Varience
```

[64]:
```
0.20391822222222222
```

[65]:
```
SE=np.mean((np.mean(ypredicted)-y)**2)
bias=SE-Varience
bias
```

[65]:
```
0.012007022222222219
```

[66]:
```
print("Confusion Matrix")
matrix = confusion_matrix(y_test,ypredicted)
print(matrix)
```

```
Confusion Matrix
[[1842  203]
 [ 302  653]]
```

[67]:
```
print("\nClassification Report")
report = classification_report(y_test,ypredicted)
print(report)
```

```

Classification Report
              precision    recall  f1-score   support
```

```
         0.0       0.86      0.90      0.88      2045
         1.0       0.76      0.68      0.72       955

    accuracy                           0.83      3000
   macro avg       0.81      0.79      0.80      3000
weighted avg       0.83      0.83      0.83      3000
```

[68]:
```python
lr_accuracy = accuracy_score(y_test, ypredicted)
lr_accuracy
print('Logistic Regression Accuracy of Scikit Model: {:.2f}%'.
 ↪format(lr_accuracy*100))
```

```
Logistic Regression Accuracy of Scikit Model: 83.17%
```

[69]:
```python
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedStratifiedKFold
```

[70]:
```python
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l2']
# define models and parameters
model = LogisticRegression()
c_values = [100, 1000, 1.0, 0.1, 0.01]
# define grid search
grid = dict(solver=solvers,penalty=penalty,C=c_values)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv,␣
 ↪scoring='accuracy',error_score=0)
grid_result = grid_search.fit(x_train,y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean*100, stdev*100, param))
```

```
Best: 0.845333 using {'C': 100, 'penalty': 'l2', 'solver': 'newton-cg'}
84.533333 (1.070275) with: {'C': 100, 'penalty': 'l2', 'solver': 'newton-cg'}
80.647619 (2.359734) with: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
80.528571 (1.225661) with: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'}
84.519048 (1.053974) with: {'C': 1000, 'penalty': 'l2', 'solver': 'newton-cg'}
80.819048 (2.513618) with: {'C': 1000, 'penalty': 'l2', 'solver': 'lbfgs'}
80.528571 (1.225661) with: {'C': 1000, 'penalty': 'l2', 'solver': 'liblinear'}
84.509524 (0.994326) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
80.833333 (2.426914) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'}
80.528571 (1.225661) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
84.390476 (0.895137) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'}
```

```
81.076190 (2.496728) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
80.528571 (1.233407) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
83.533333 (0.972362) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}
80.600000 (1.885642) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
80.509524 (1.139688) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}
```

[71]:
```
grid_result.score(x_train,y_train)*100
```

[71]:
```
84.81428571428572
```

[73]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import cross_val_score
```

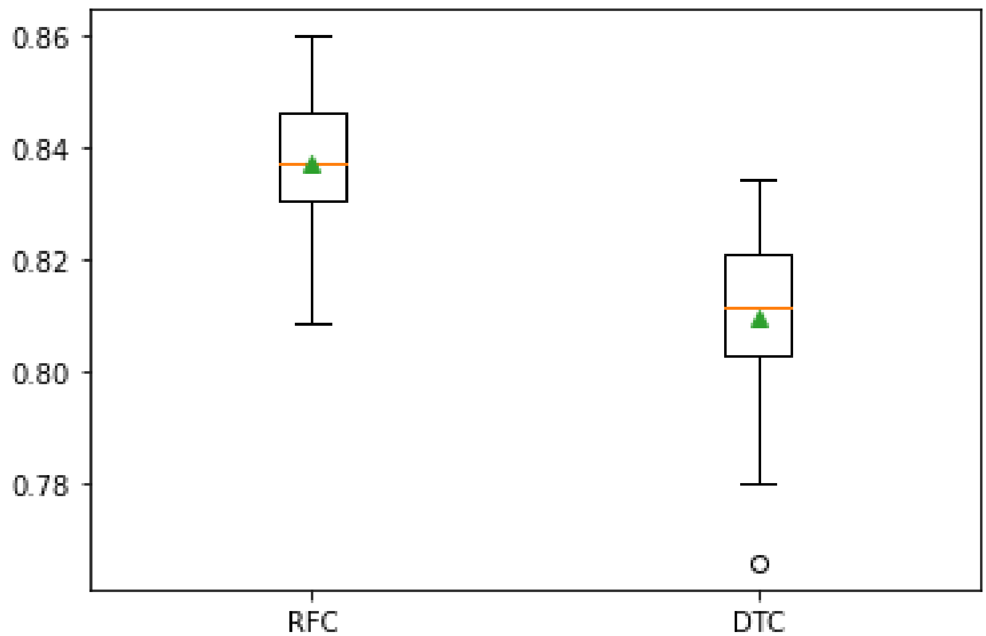
[75]:
```
# Spot-Check Algorithms
models = []
models.append(('RF', RandomForestClassifier()))
models.append(('GNB', GaussianNB()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DSC', DecisionTreeClassifier(random_state = 1, max_depth=2)))
models.append(('SVM', SVC()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = RepeatedStratifiedKFold(n_splits=10, n_repeats = 3, random_state=1)
    cv_results = cross_val_score(model,x_train,y_train, cv=kfold,␣
 ↪scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %.2f (%.3f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
RF: 0.84 (0.011)
GNB: 0.73 (0.017)
KNN: 0.79 (0.014)
DSC: 0.81 (0.015)
SVM: 0.70 (0.004)
```

[76]:
```
# evaluate model 1
model1 = RandomForestClassifier()
cv1 = RepeatedStratifiedKFold(n_splits = 10, n_repeats = 3, random_state = 1)
scores1 = cross_val_score(model1,x_train,y_train, scoring = 'accuracy', cv =␣
 ↪cv1, n_jobs = -1)
```

```
print('RFC Mean Accuracy: %.1f%% +/-(%.3f)' % (np.mean(scores1*100), np.
↪std(scores1)))
# evaluate model 2
model2 = DecisionTreeClassifier(random_state = 1, max_depth=2)
cv2 = RepeatedStratifiedKFold(n_splits = 10, n_repeats = 3, random_state = 1)
scores3 = cross_val_score(model2,x_train,y_train, scoring = 'accuracy', cv =␣
↪cv2, n_jobs = -1)
print('DecisionTreeClassifier Mean Accuracy: %.1f%% +/-(%.3f)' % (np.
↪mean(scores3*100), np.std(scores3)))
# plot the results
plt.boxplot([scores1, scores3], labels=['RFC', 'DTC'], showmeans=True)
plt.show()
```

```
RFC Mean Accuracy: 83.7% +/-(0.012)
DecisionTreeClassifier Mean Accuracy: 80.9% +/-(0.015)
```



[77]:
```
from mlxtend.evaluate import paired_ttest_5x2cv
# check if difference between algorithms is real
t, p = paired_ttest_5x2cv(estimator1=model1,
                          estimator2=model2,
                          X=x,
                          y=y,
                          scoring='accuracy',
                          random_seed=1)
# summarize
```

```python
print(f'The P-value is = {p:.3f}')
print(f'The t-statistics is = {t:.3f}')
# interpret the result
if p <= 0.05:
    print('Since p<0.05, We can reject the null-hypothesis that both models␣
 ↪perform equally well on this dataset. We may conclude that the two␣
 ↪algorithms are significantly different.')
else:
    print('Since p>0.05, we cannot reject the null hypothesis and may conclude␣
 ↪that the performance of the two algorithms is not significantly different.')
```

```
The P-value is = 0.021
The t-statistics is = 3.339
Since p<0.05, We can reject the null-hypothesis that both models perform equally
well on this dataset. We may conclude that the two algorithms are significantly
different.
```

[ ]:

```

```