In [1]:
```
#This project focuses on testing ensemble methods to increase accuracy of models to
```

In [2]:
```
#Import Libraries

#preliminary data analysis
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math

#Feature Scaling
from sklearn.preprocessing import MinMaxScaler

#OneHot Encoding categorical data into numerical
from sklearn.preprocessing import OneHotEncoder

#Split the Data
from sklearn.model_selection import train_test_split

#Balance the Data - balance only the training dataset
#Import Libraries for balancing
from imblearn.over_sampling import SMOTE
from imblearn.combine import SMOTEENN
from imblearn.under_sampling import EditedNearestNeighbours

#Develop the Models
from sklearn import metrics
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    f1_score,
)
from sklearn.model_selection import cross_val_score, KFold, cross_val_predict
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV

#The Models
from sklearn.naive_bayes import CategoricalNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
#!pip install keras
#!pip install --upgrade h5py
#!pip install TensorFlow
#!pip install scikeras

import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

```
from keras.callbacks import EarlyStopping
from keras import optimizers
from scikeras.wrappers import KerasClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification

#Ensemble Learning
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import StackingClassifier

from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
```

In [3]:
```
#housekeeping
pd.set_option('display.max_columns', None) #allow to see all columns when looking a
```

In [4]:
```
#THE DATA
df=pd.read_csv("C:\\Users\\carol\\Desktop\\School\\Ryerson\\CIND860\\insurance_clai
```

In [5]:
```
##Exploring the Data
```

In [6]:
```
df.head()
```

Out[6]:

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	poli
0	328	48	521585	2014-10-17	OH	250/500	
1	228	42	342868	2006-06-27	IN	250/500	
2	134	29	687698	2000-09-06	OH	100/300	
3	256	41	227811	1990-05-25	IL	250/500	
4	228	44	367455	2014-06-06	IL	500/1000	

In [7]:
```
df.describe()
```

Out[7]:

	months_as_customer	age	policy_number	policy_deductable	policy_annual_p
count	1000.000000	1000.000000	1000.000000	1000.000000	100
mean	203.954000	38.948000	546238.648000	1136.000000	125
std	115.113174	9.140287	257063.005276	611.864673	24
min	0.000000	19.000000	100804.000000	500.000000	43
25%	115.750000	32.000000	335980.250000	500.000000	108
50%	199.500000	38.000000	533135.000000	1000.000000	125
75%	276.250000	44.000000	759099.750000	2000.000000	141
max	479.000000	64.000000	999435.000000	2000.000000	204

In [8]:
```
df.info()
```

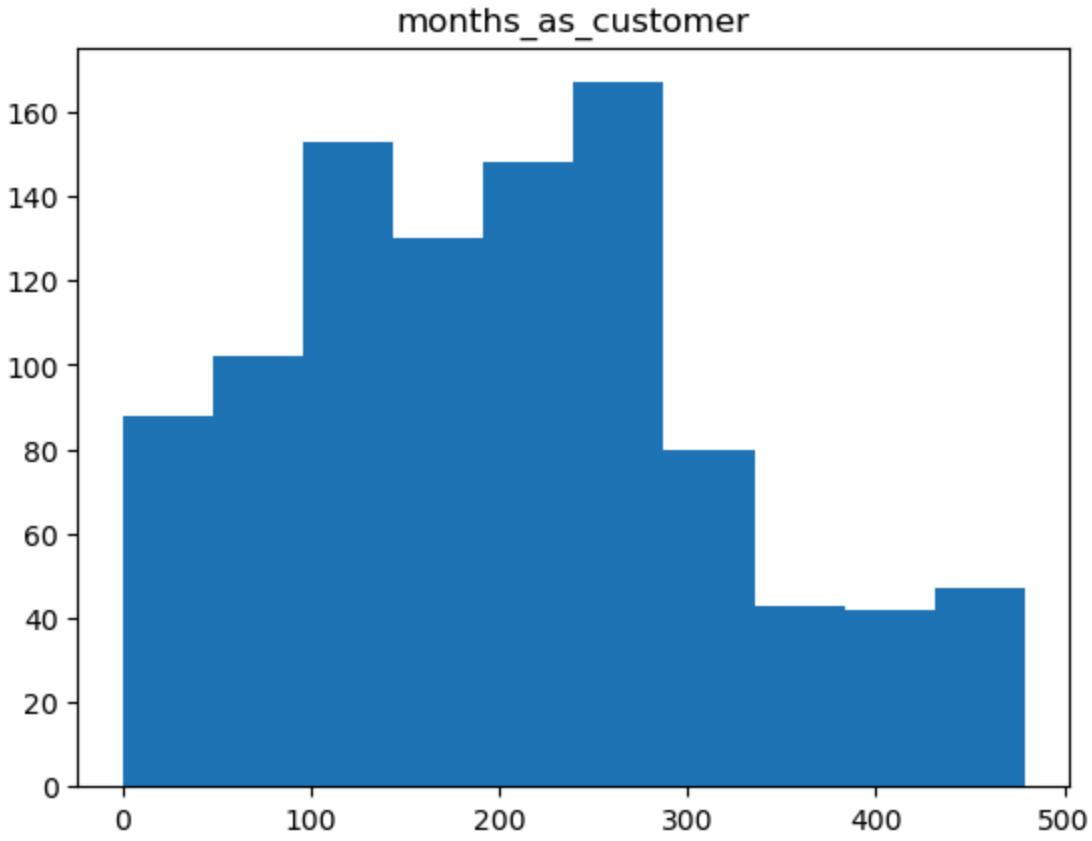
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 40 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   months_as_customer           1000 non-null   int64  
 1   age                          1000 non-null   int64  
 2   policy_number                1000 non-null   int64  
 3   policy_bind_date             1000 non-null   object 
 4   policy_state                 1000 non-null   object 
 5   policy_csl                   1000 non-null   object 
 6   policy_deductable            1000 non-null   int64  
 7   policy_annual_premium        1000 non-null   float64
 8   umbrella_limit               1000 non-null   int64  
 9   insured_zip                  1000 non-null   int64  
 10  insured_sex                  1000 non-null   object 
 11  insured_education_level      1000 non-null   object 
 12  insured_occupation           1000 non-null   object 
 13  insured_hobbies              1000 non-null   object 
 14  insured_relationship         1000 non-null   object 
 15  capital-gains                1000 non-null   int64  
 16  capital-loss                 1000 non-null   int64  
 17  incident_date                1000 non-null   object 
 18  incident_type                1000 non-null   object 
 19  collision_type               1000 non-null   object 
 20  incident_severity            1000 non-null   object 
 21  authorities_contacted        1000 non-null   object 
 22  incident_state               1000 non-null   object 
 23  incident_city                1000 non-null   object 
 24  incident_location            1000 non-null   object 
 25  incident_hour_of_the_day     1000 non-null   int64  
 26  number_of_vehicles_involved  1000 non-null   int64  
 27  property_damage              1000 non-null   object 
 28  bodily_injuries              1000 non-null   int64  
 29  witnesses                    1000 non-null   int64  
 30  police_report_available      1000 non-null   object 
 31  total_claim_amount           1000 non-null   int64  
 32  injury_claim                 1000 non-null   int64  
 33  property_claim               1000 non-null   int64  
 34  vehicle_claim                1000 non-null   int64  
 35  auto_make                    1000 non-null   object 
 36  auto_model                   1000 non-null   object 
 37  auto_year                    1000 non-null   int64  
 38  fraud_reported               1000 non-null   object 
 39  _c39                         0 non-null      float64
dtypes: float64(2), int64(17), object(21)
memory usage: 312.6+ KB
```

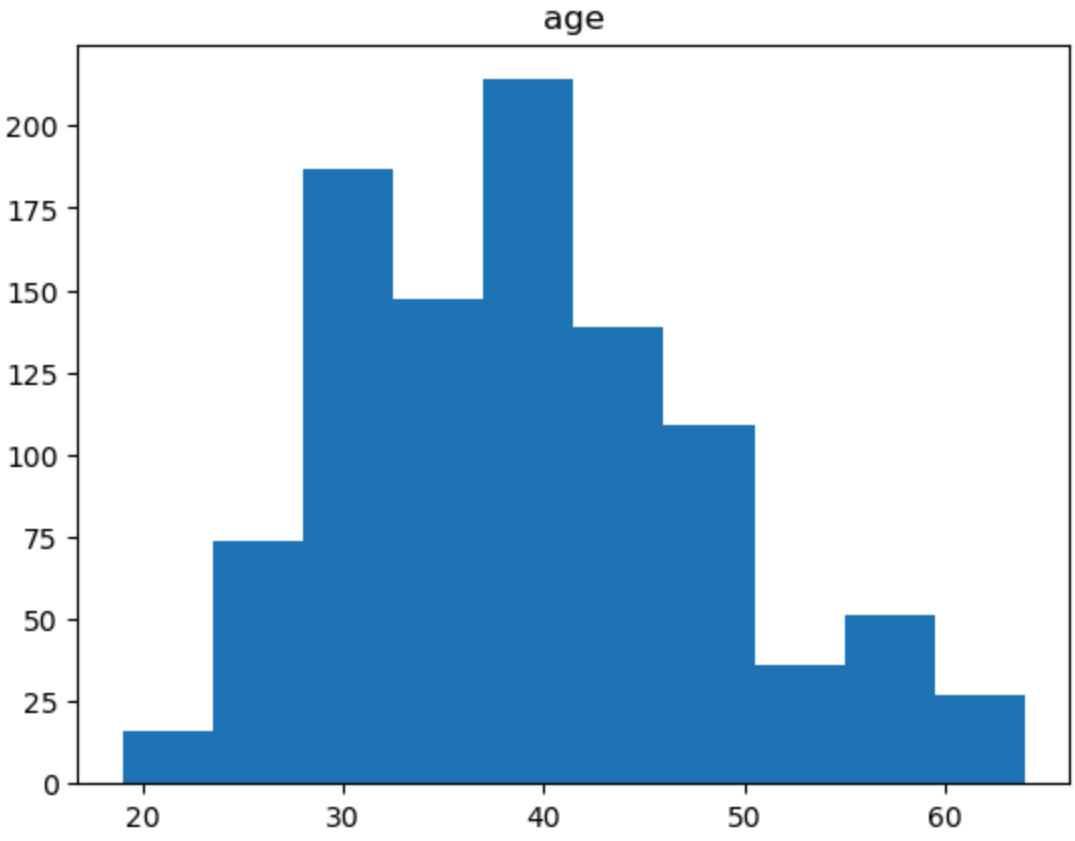
In [9]:
```
df = df.drop("_c39", axis=1) # junk column
```

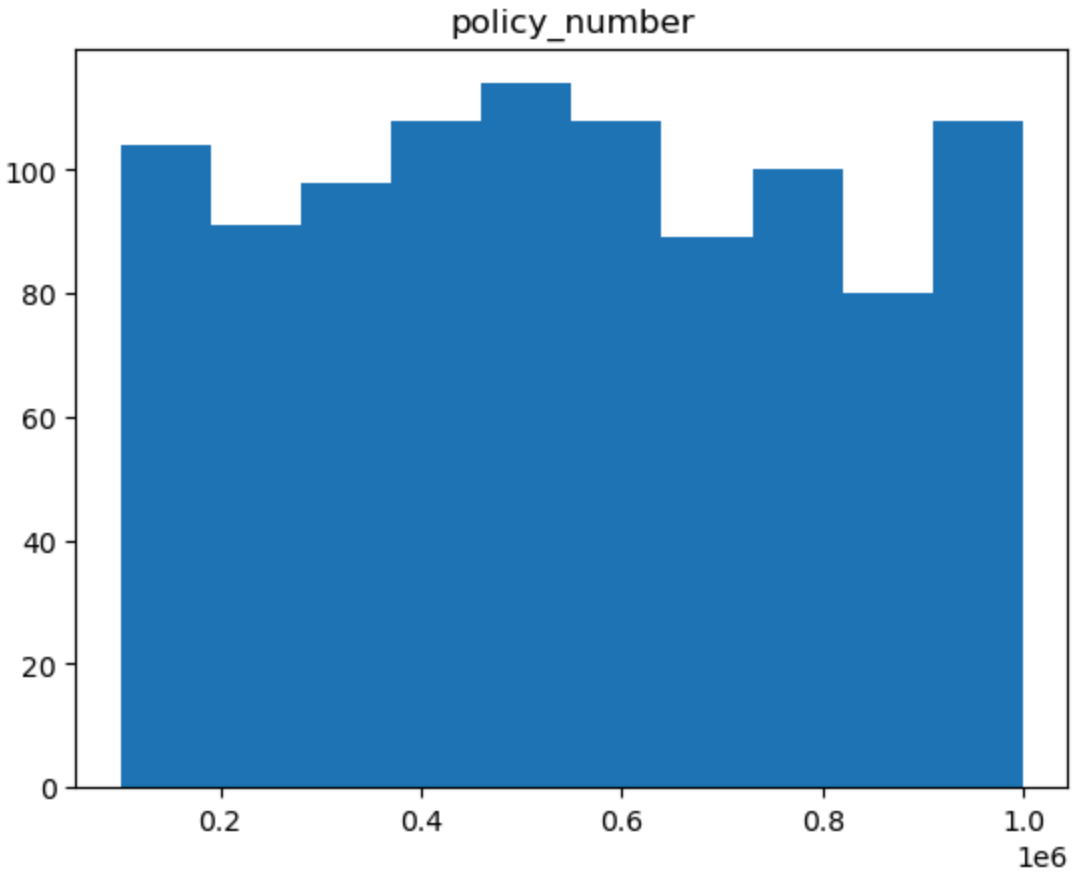
In [ ]:
```

```

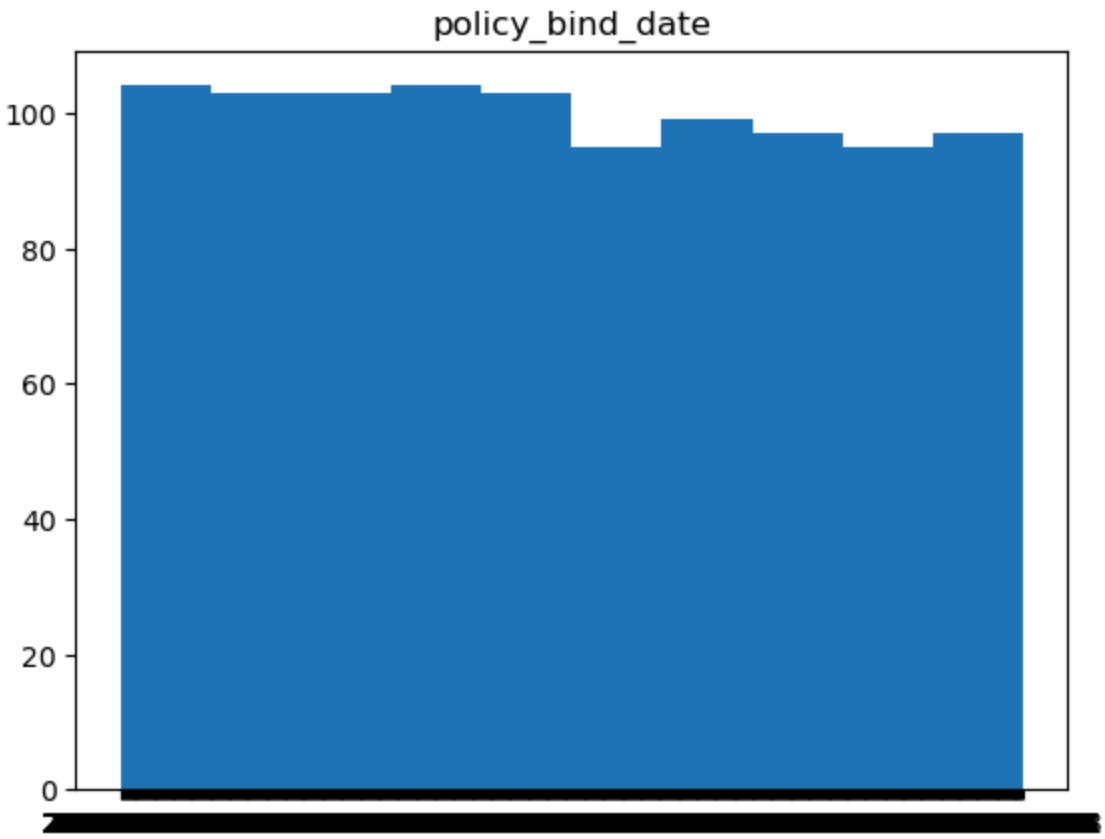
In [10]:
```
#plot the quantitative features to visualize and see whether there is normal distri
for i in df.columns:
    plt.hist(df[i])
```

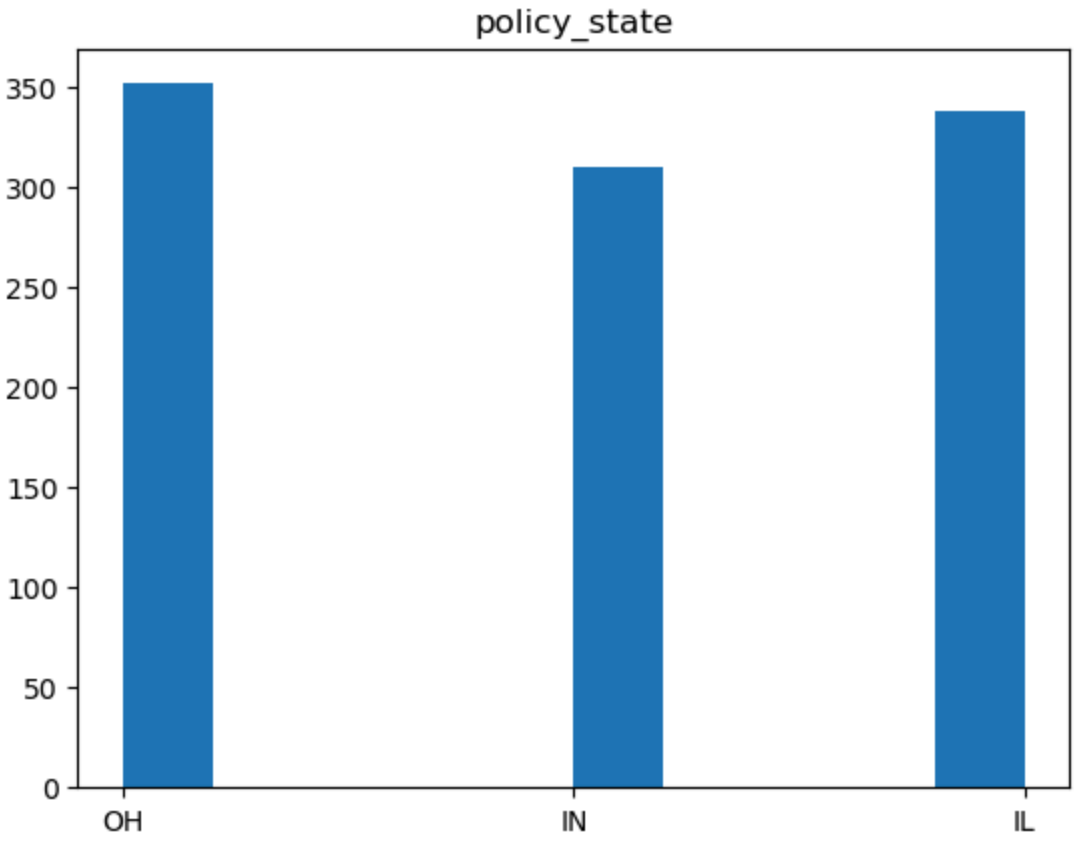
```
    plt.title(i)
    plt.show()
```

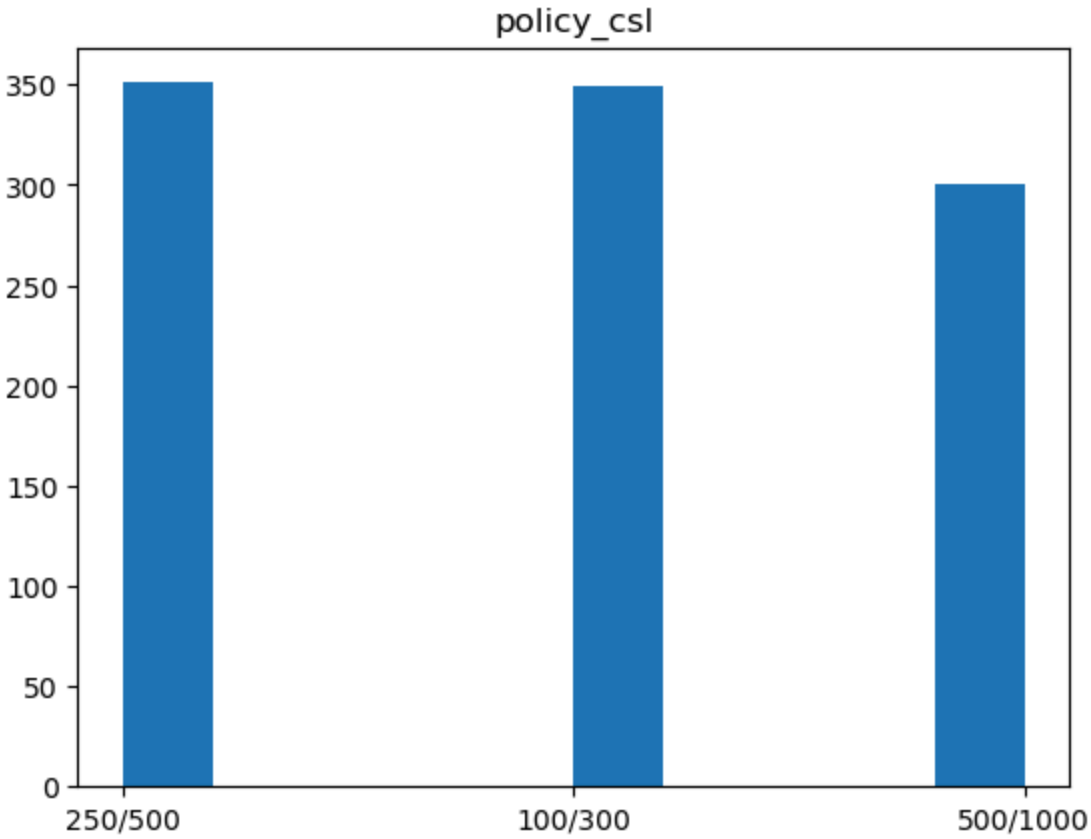


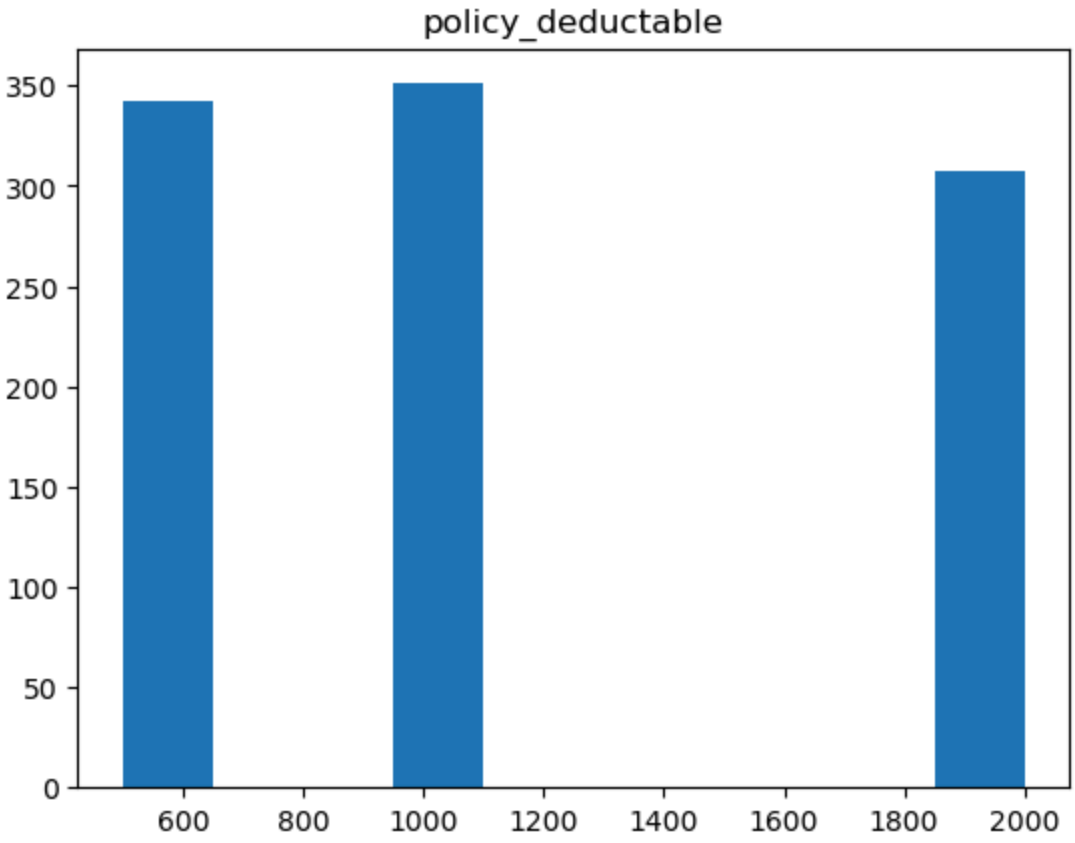


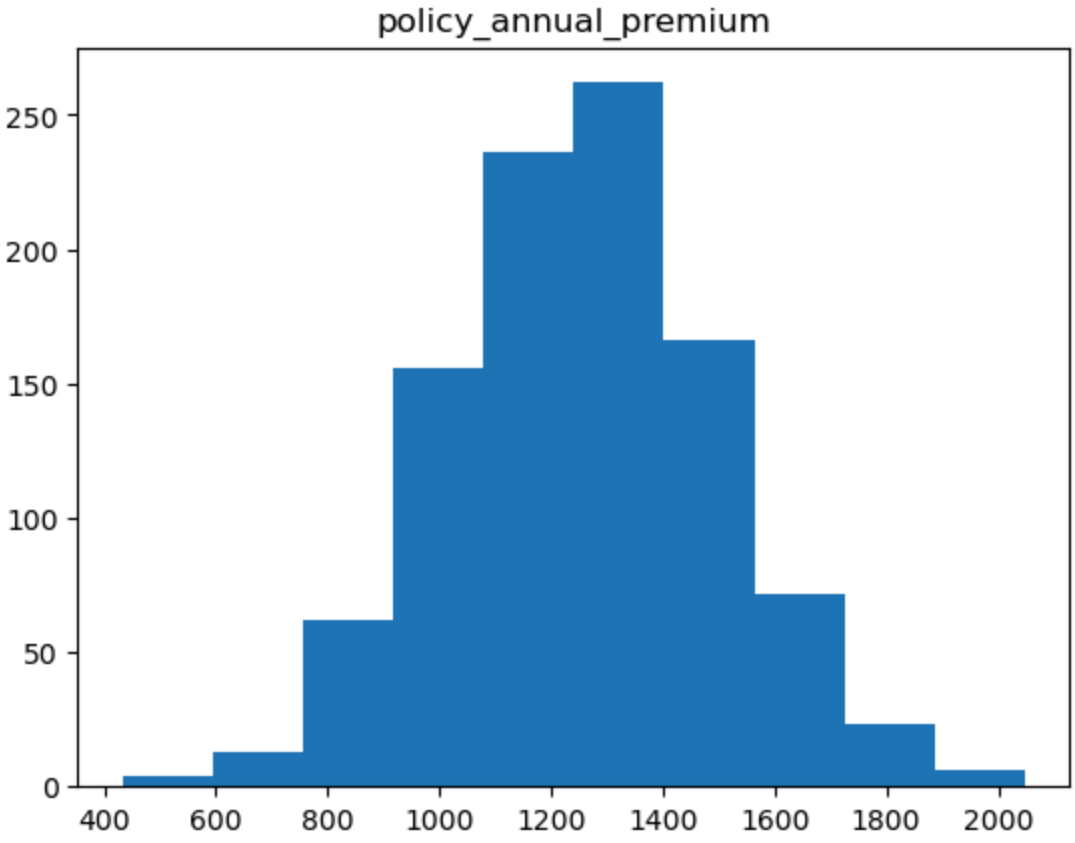

policy_number
100
80
60
40
20
0
0.2
0.4
0.6
0.8
1.0
1e6

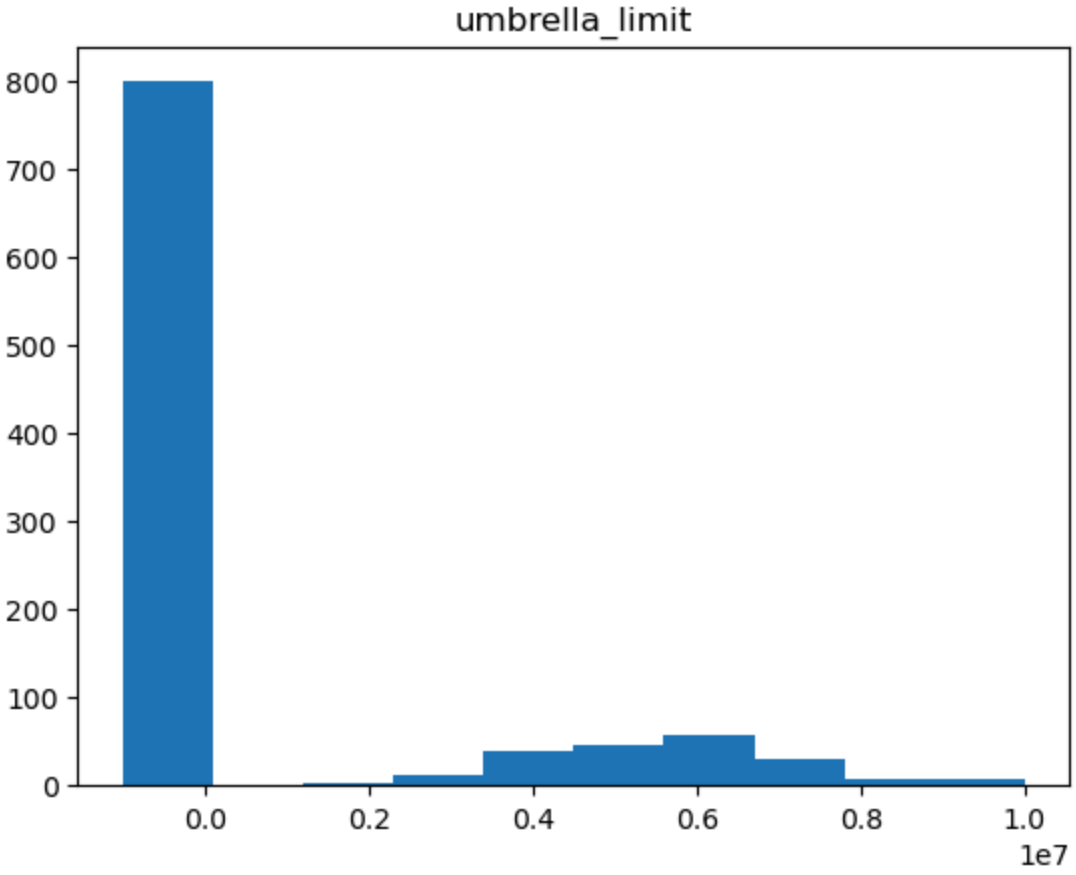

policy_bind_date
100
80
60
40
20
0

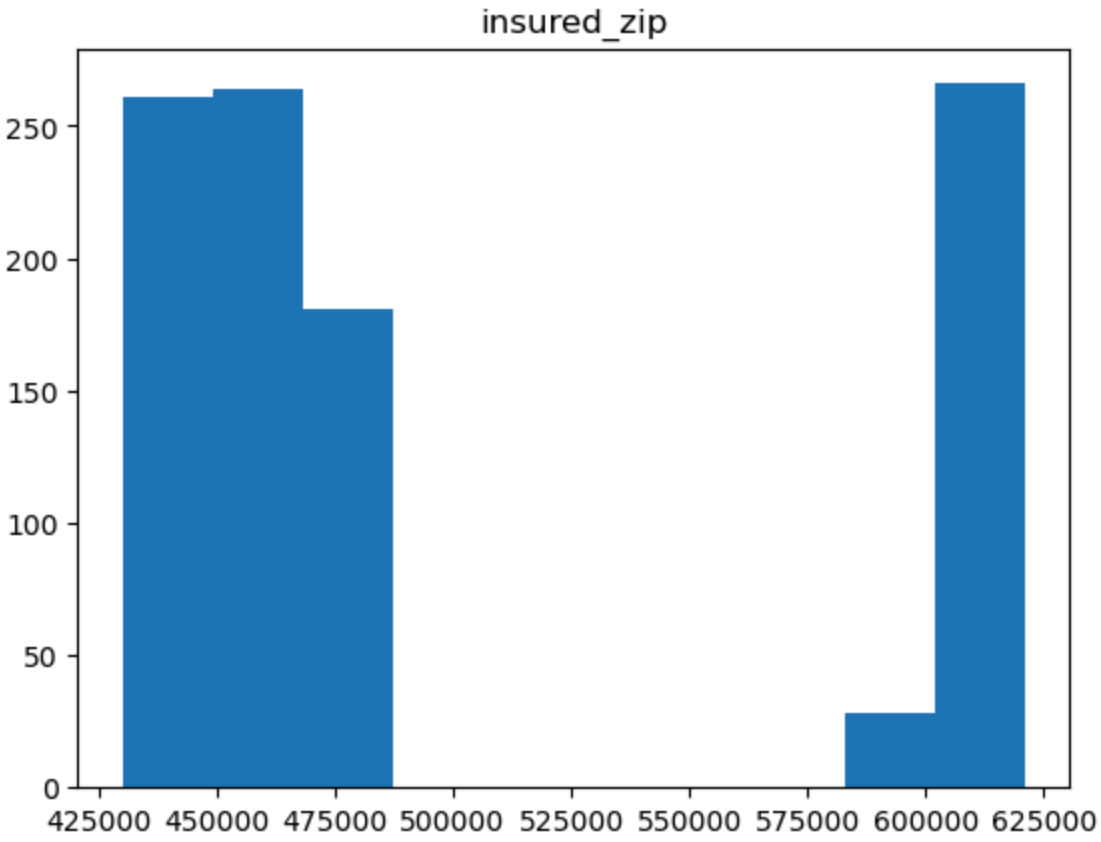

policy_state
350
300
250
200
150
100
50
0
OH
IN
IL

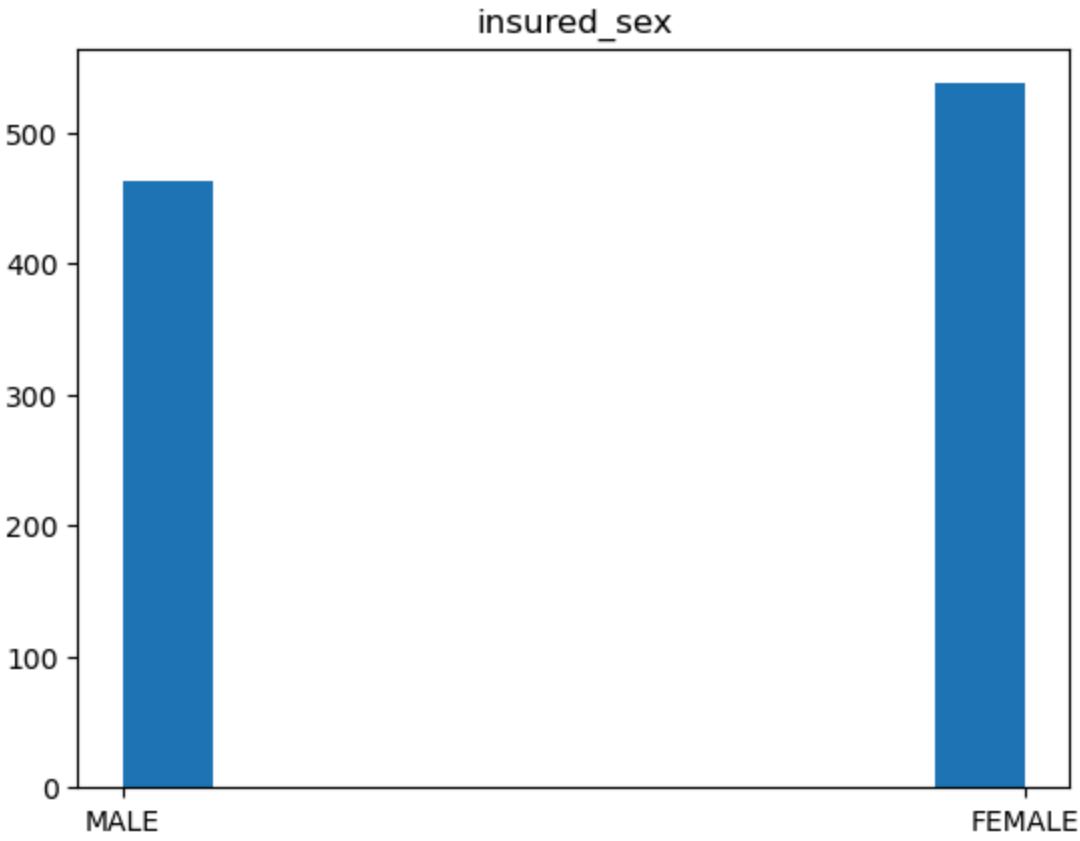

policy_csl
350
300
250
200
150
100
50
0
250/500
100/300
500/1000

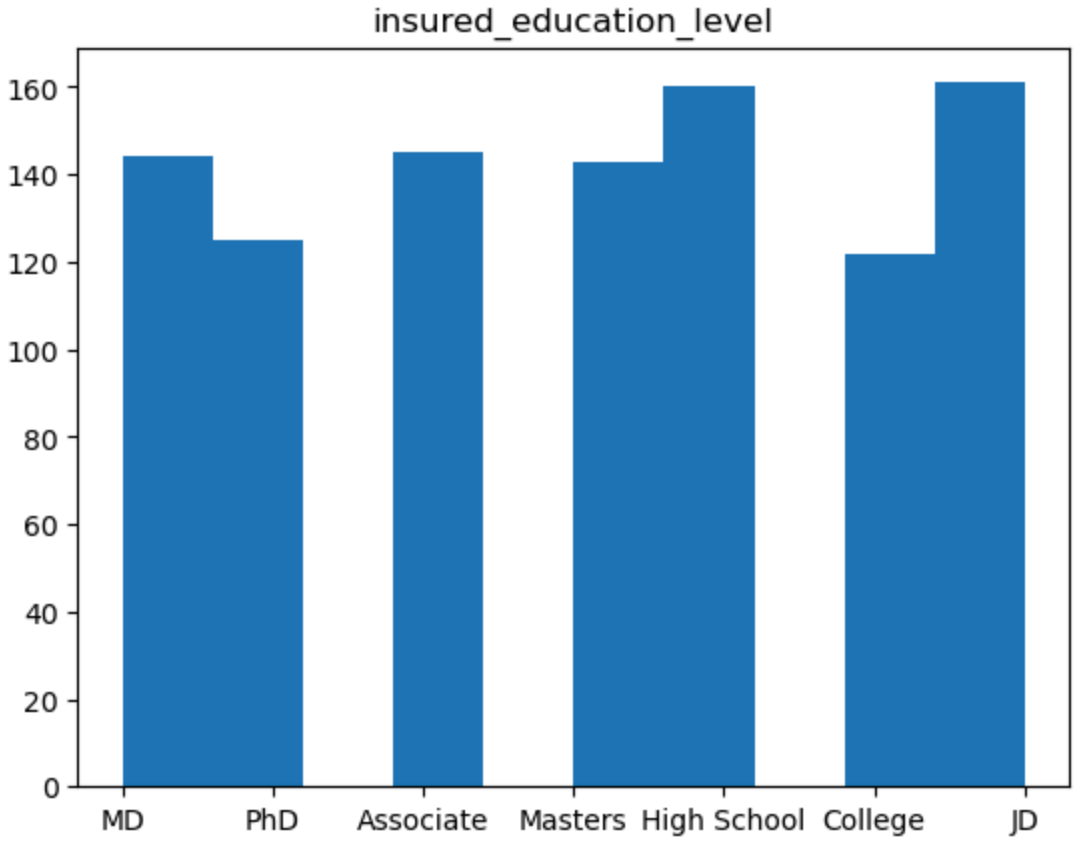

policy_deductable
350
300
250
200
150
100
50
0
600
800
1000
1200
1400
1600
1800
2000


policy_annual_premium
250
200
150
100
50
0
400
600
800
1000
1200
1400
1600
1800
2000

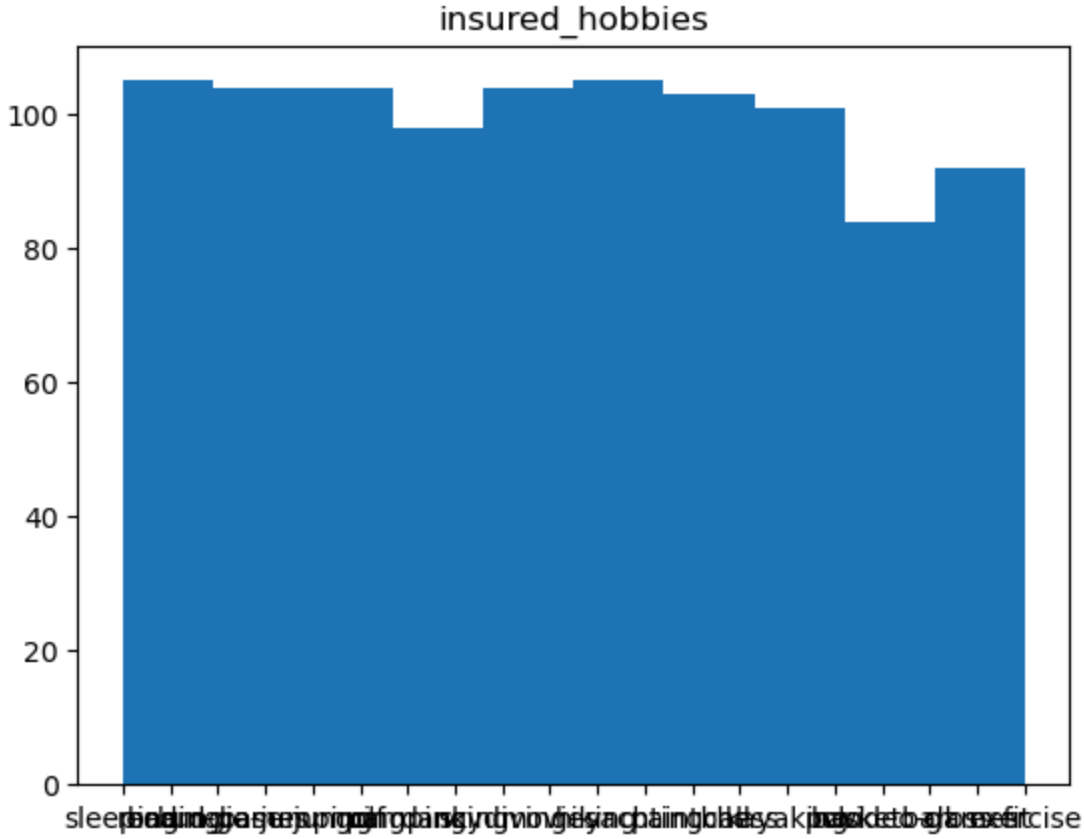

umbrella_limit
800
700
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
1e7

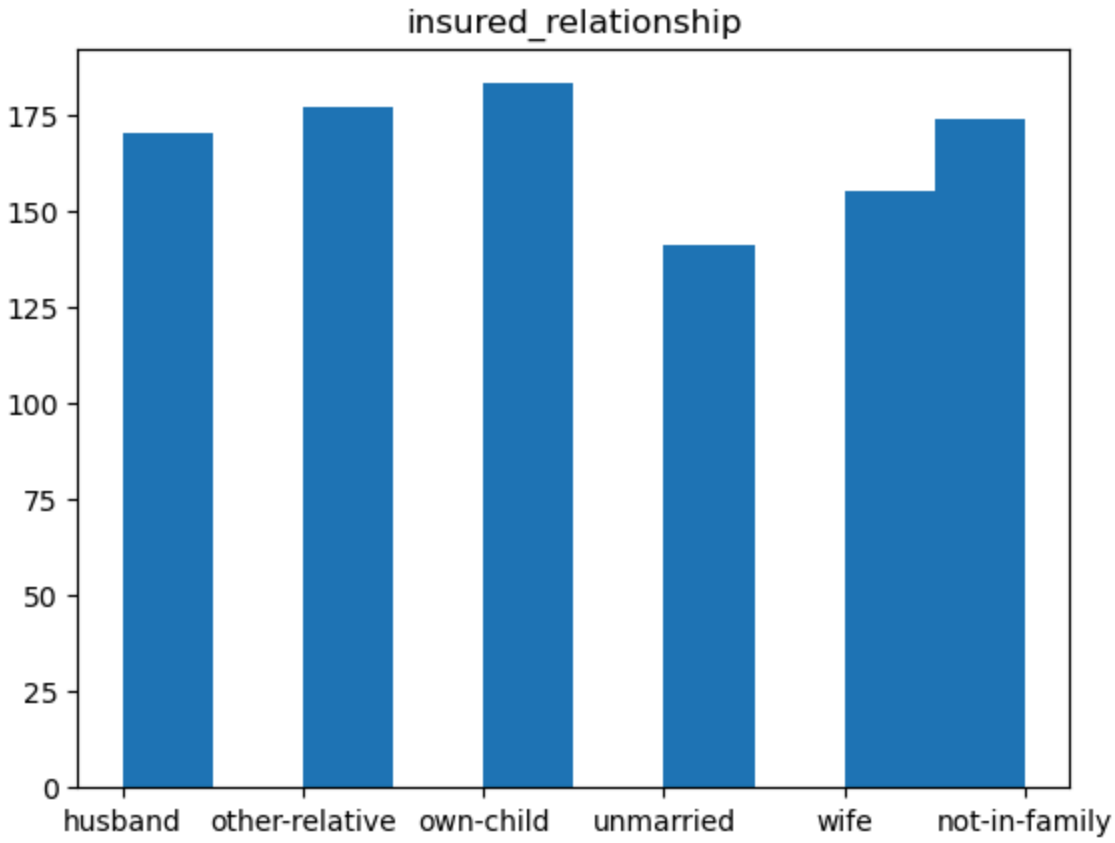

insured_zip
250
200
150
100
50
0
425000
450000
475000
500000
525000
550000
575000
600000
625000

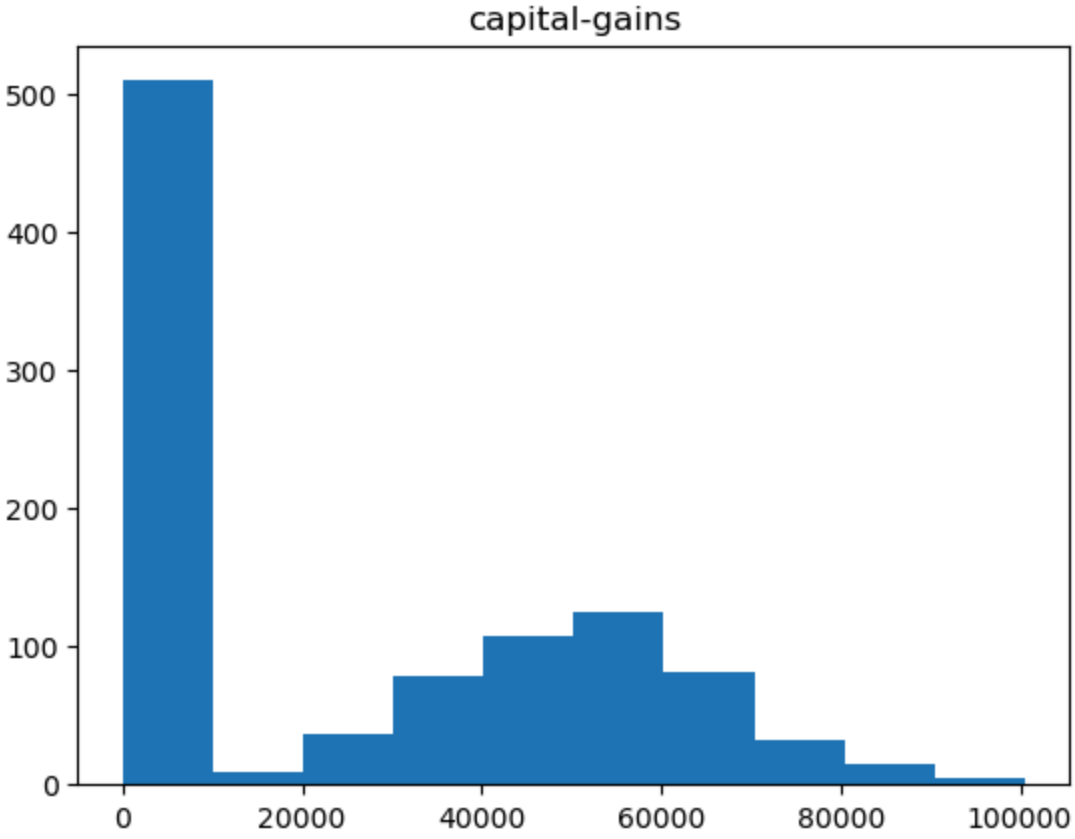

insured_sex
500
400
300
200
100
0
MALE
FEMALE

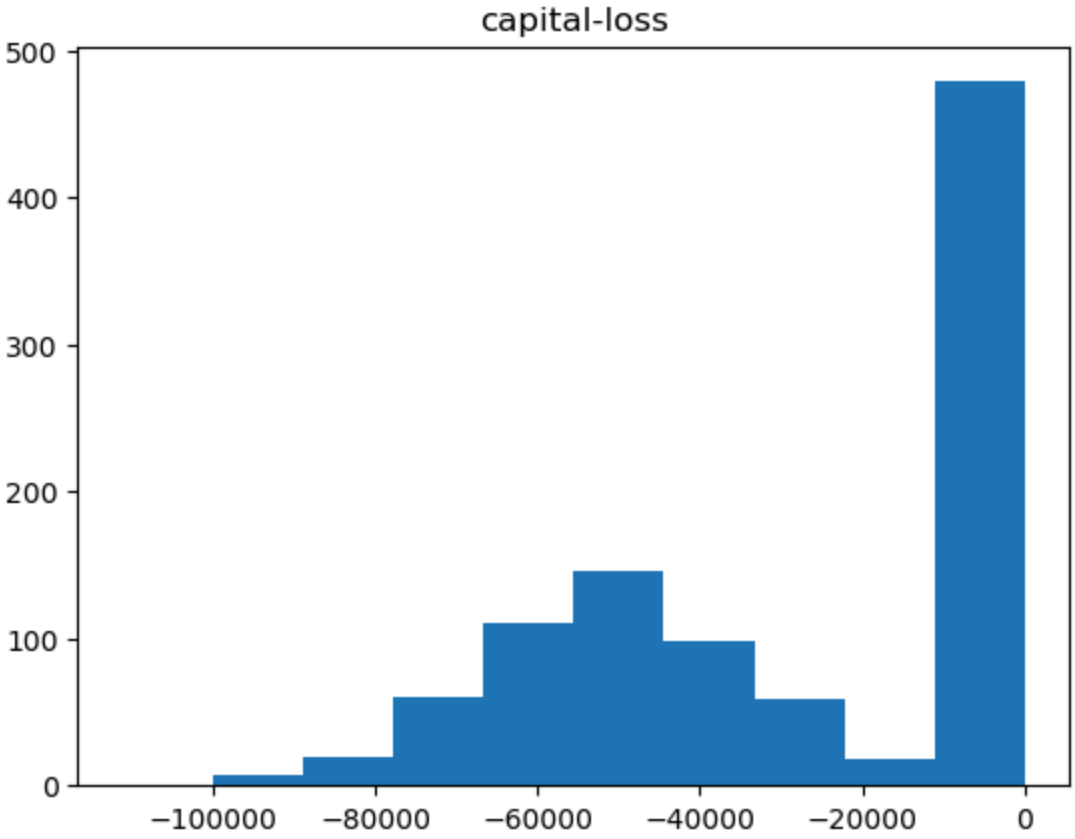

insured_education_level
160
140
120
100
80
60
40
20
0
MD
PhD
Associate
Masters
High School
College
JD

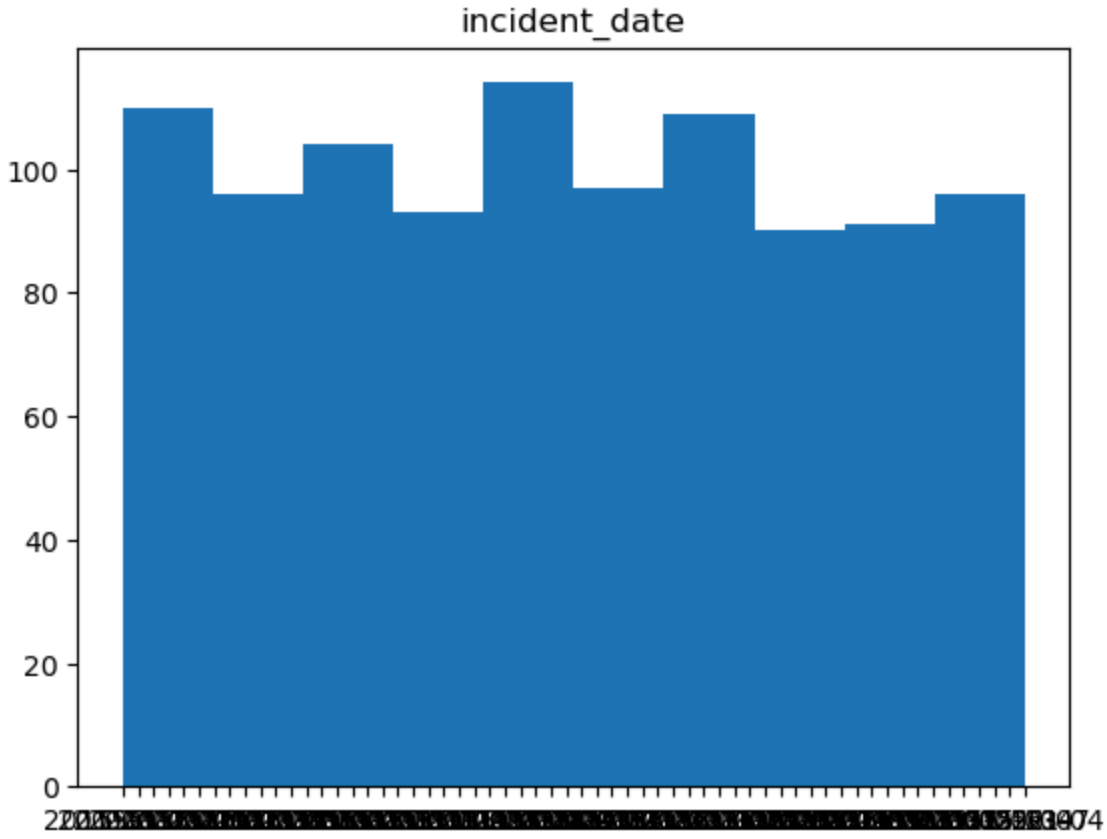

insured_occupation
160
140
120
100
80
60
40
20
0
fishing

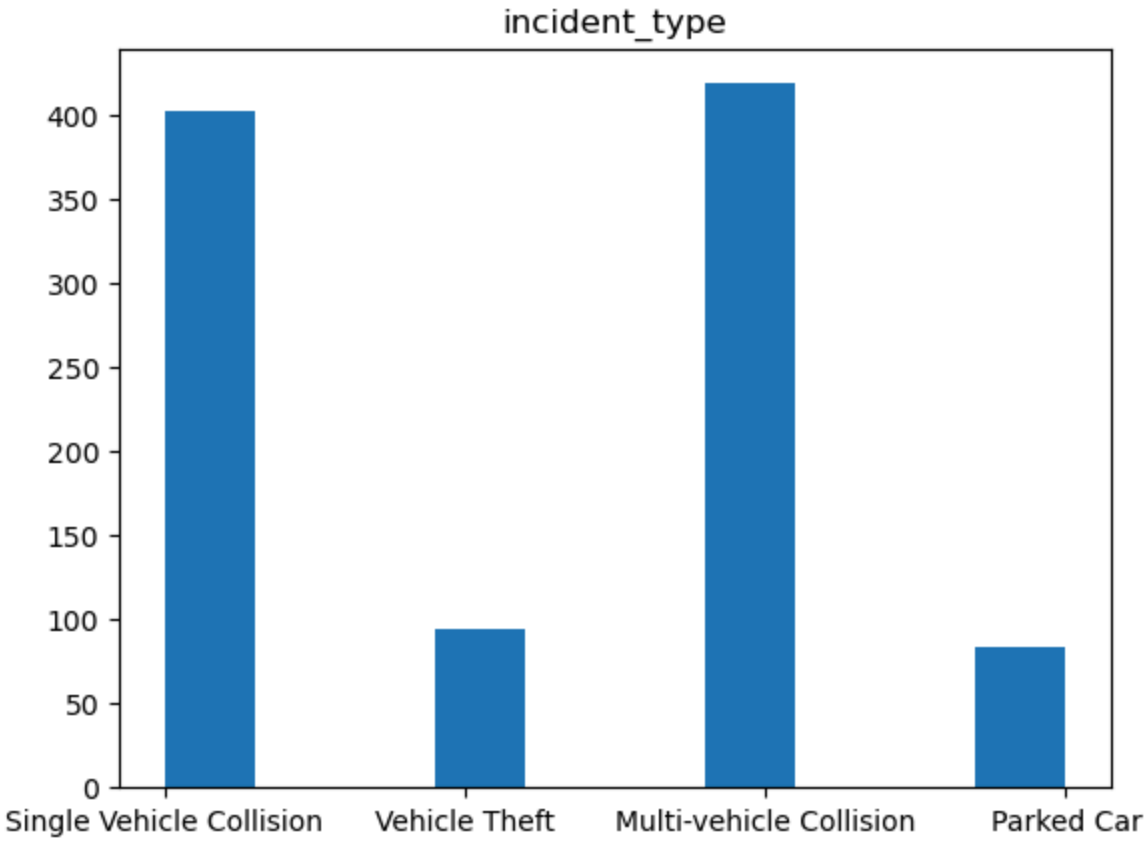

insured_hobbies
100
80
60
40
20
0
sleep
exercise

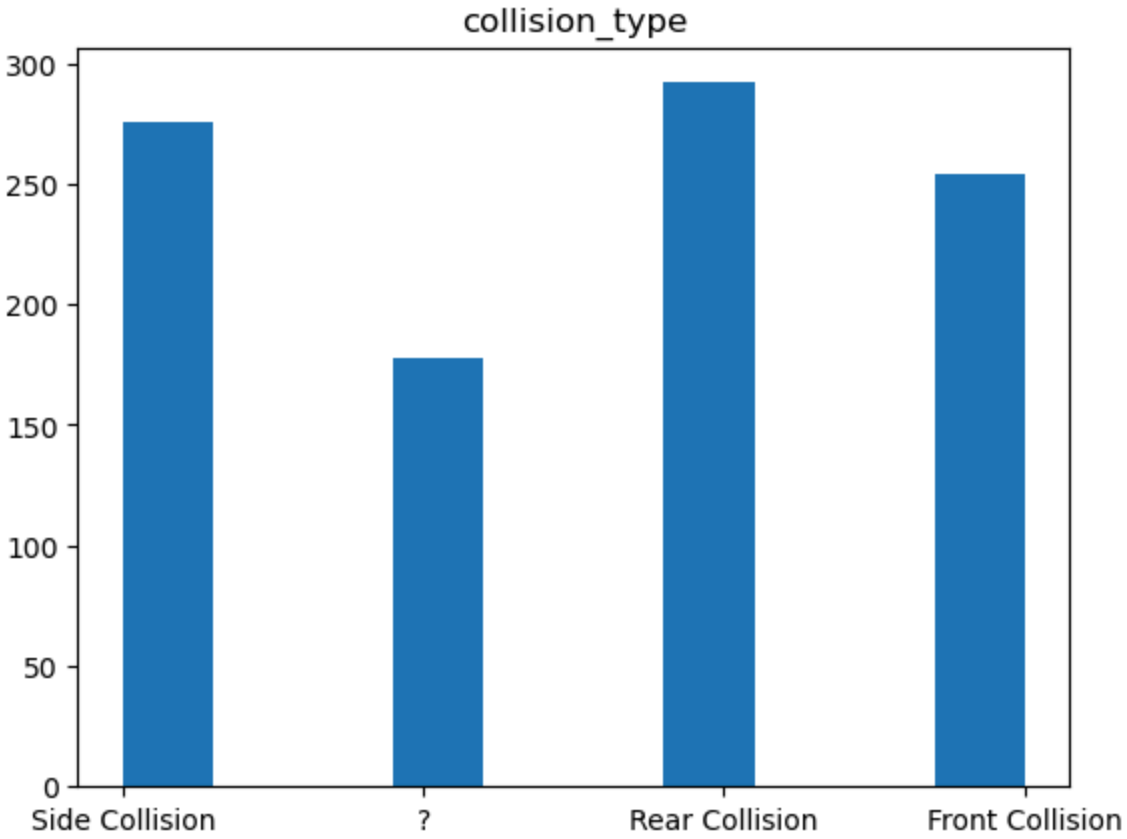

insured_relationship
175
150
125
100
75
50
25
0
husband
other-relative
own-child
unmarried
wife
not-in-family

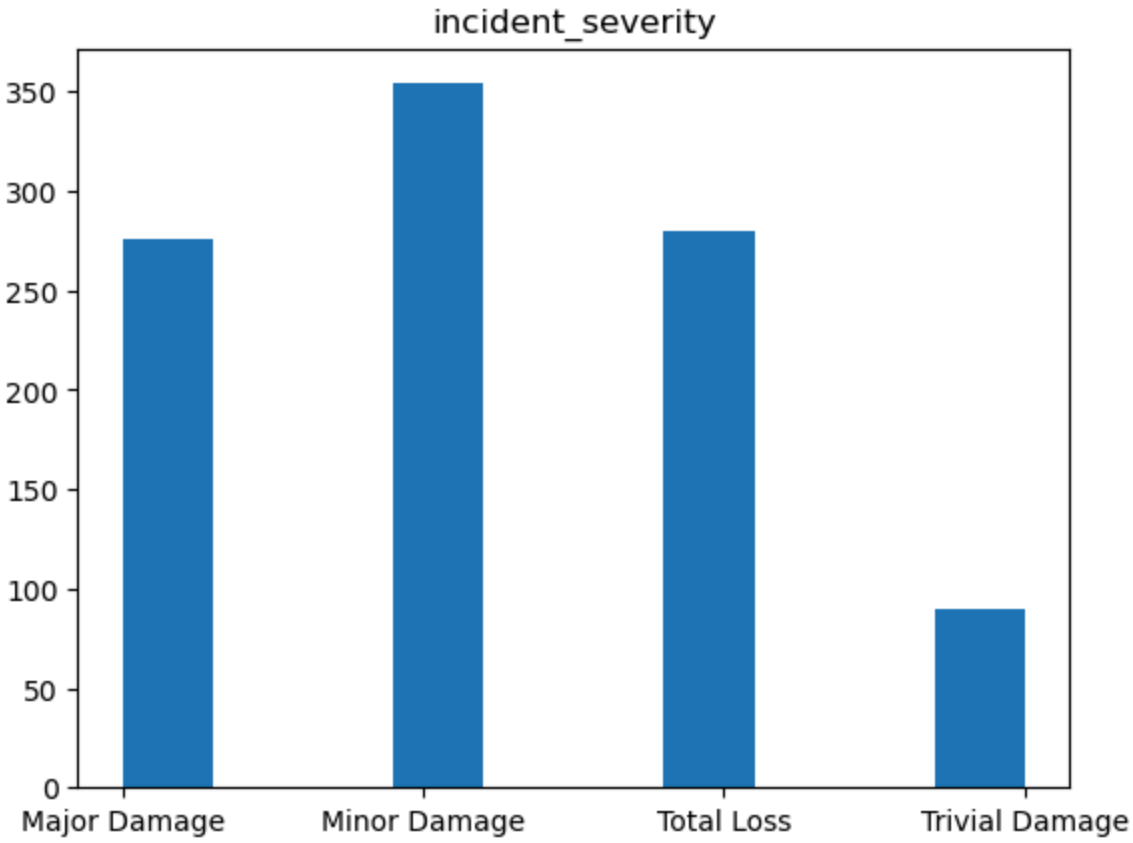

capital-gains
500
400
300
200
100
0
0
20000
40000
60000
80000
100000

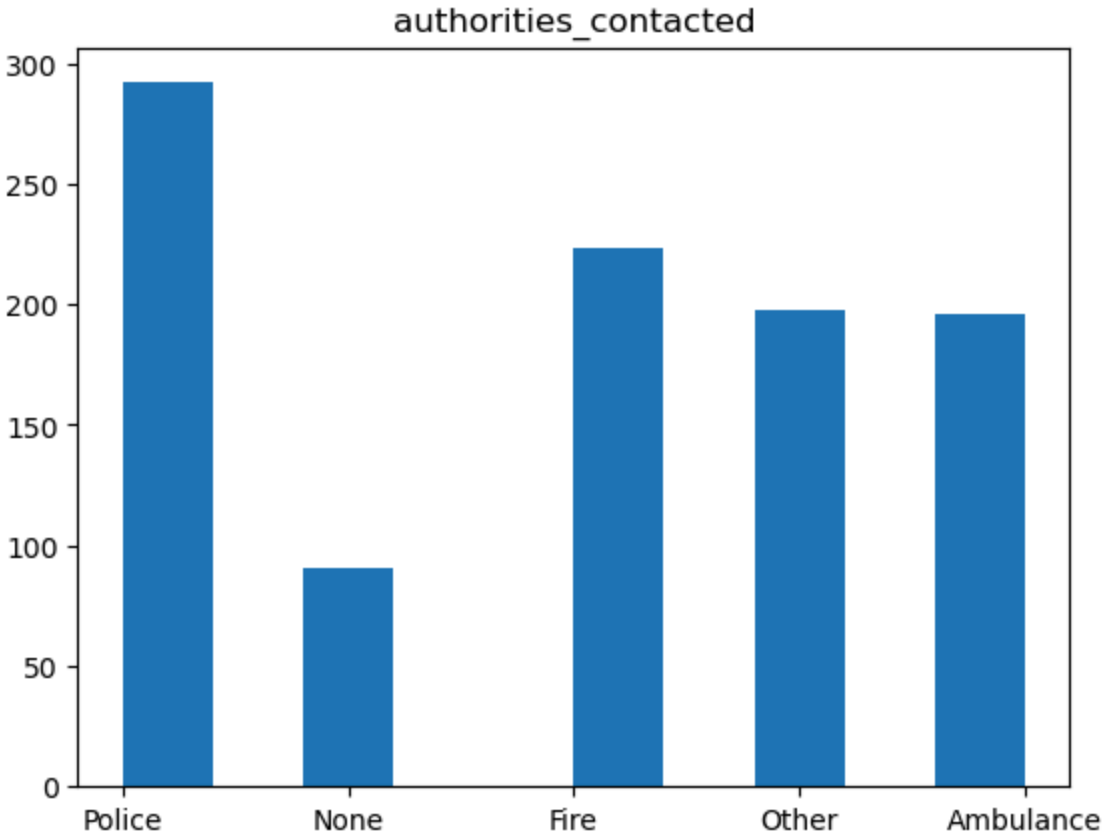

capital-loss
500
400
300
200
100
0
−100000
−80000
−60000
−40000
−20000
0

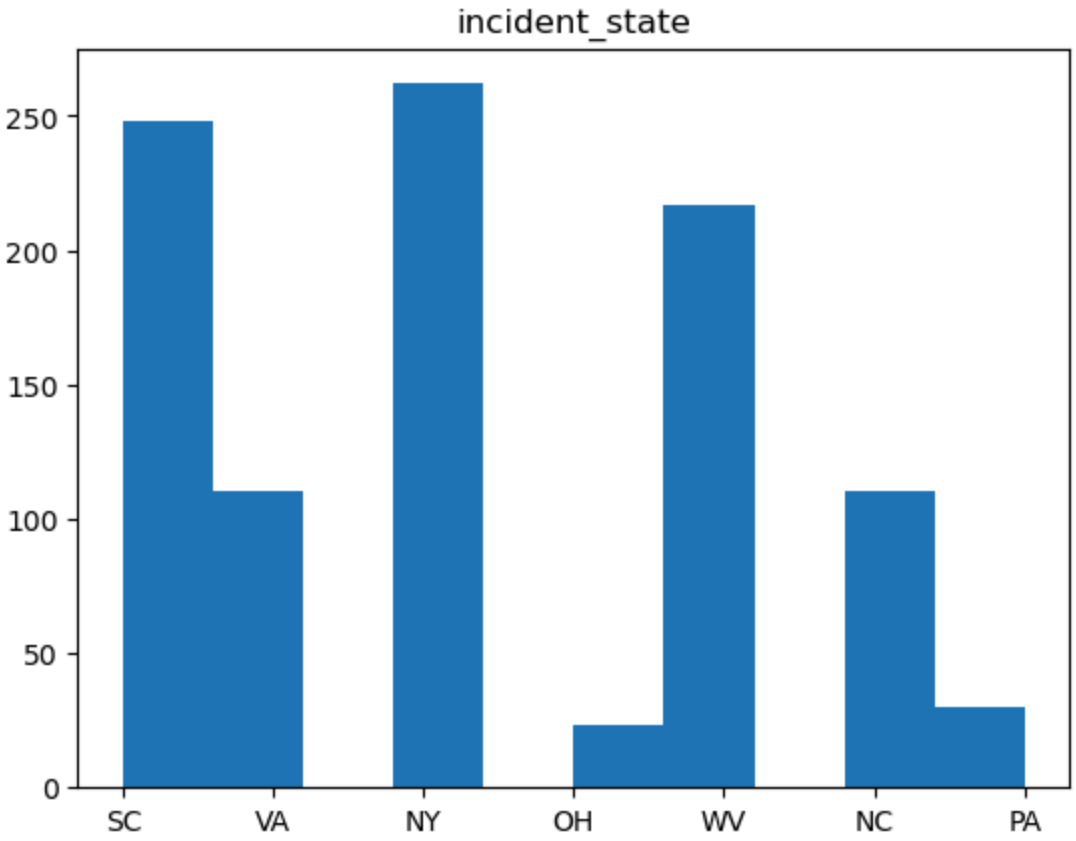

incident_date
100
80
60
40
20
0

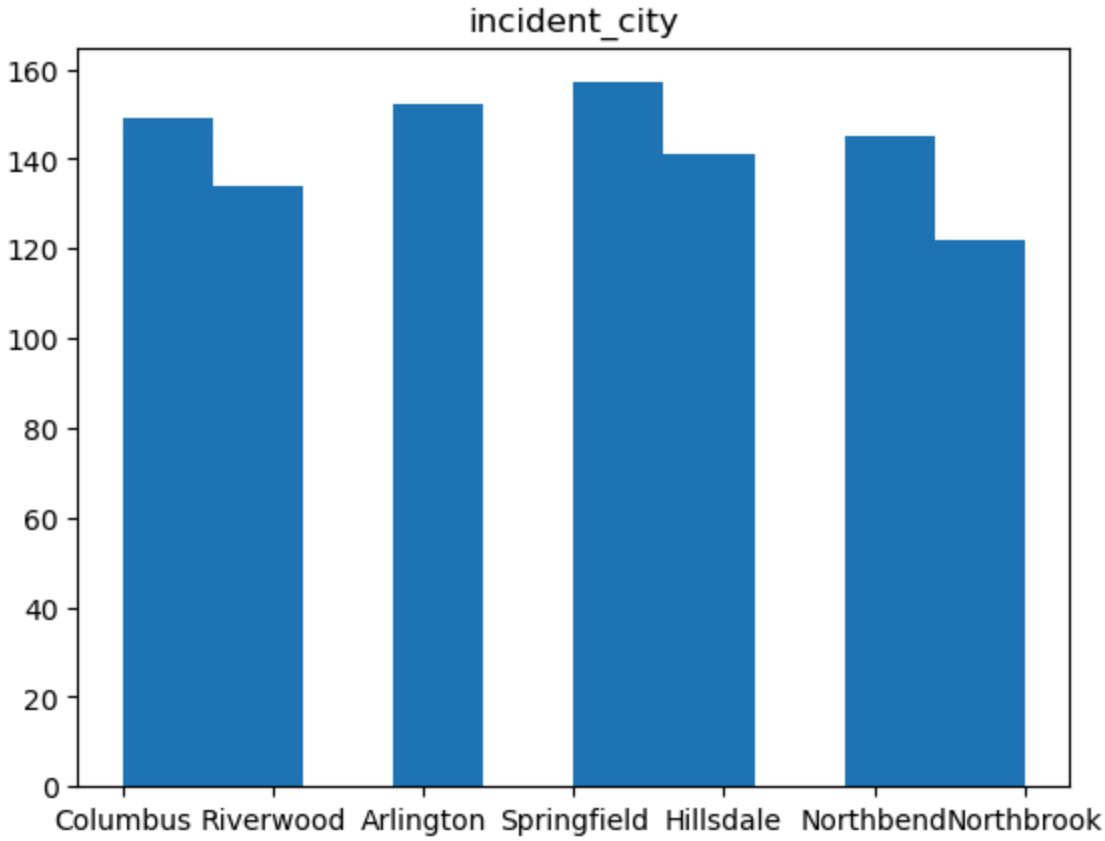

incident_type
400
350
300
250
200
150
100
50
0
Single Vehicle Collision
Vehicle Theft
Multi-vehicle Collision
Parked Car

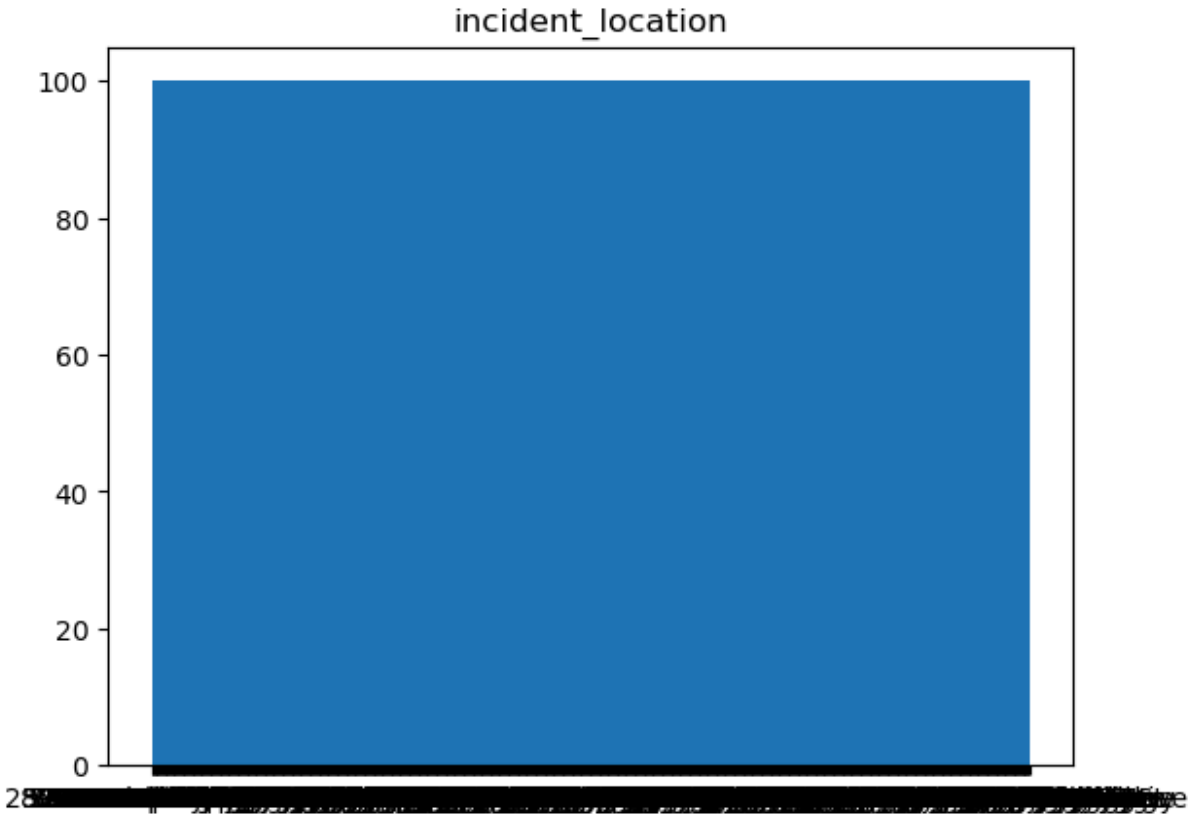

collision_type
300
250
200
150
100
50
0
Side Collision
?
Rear Collision
Front Collision

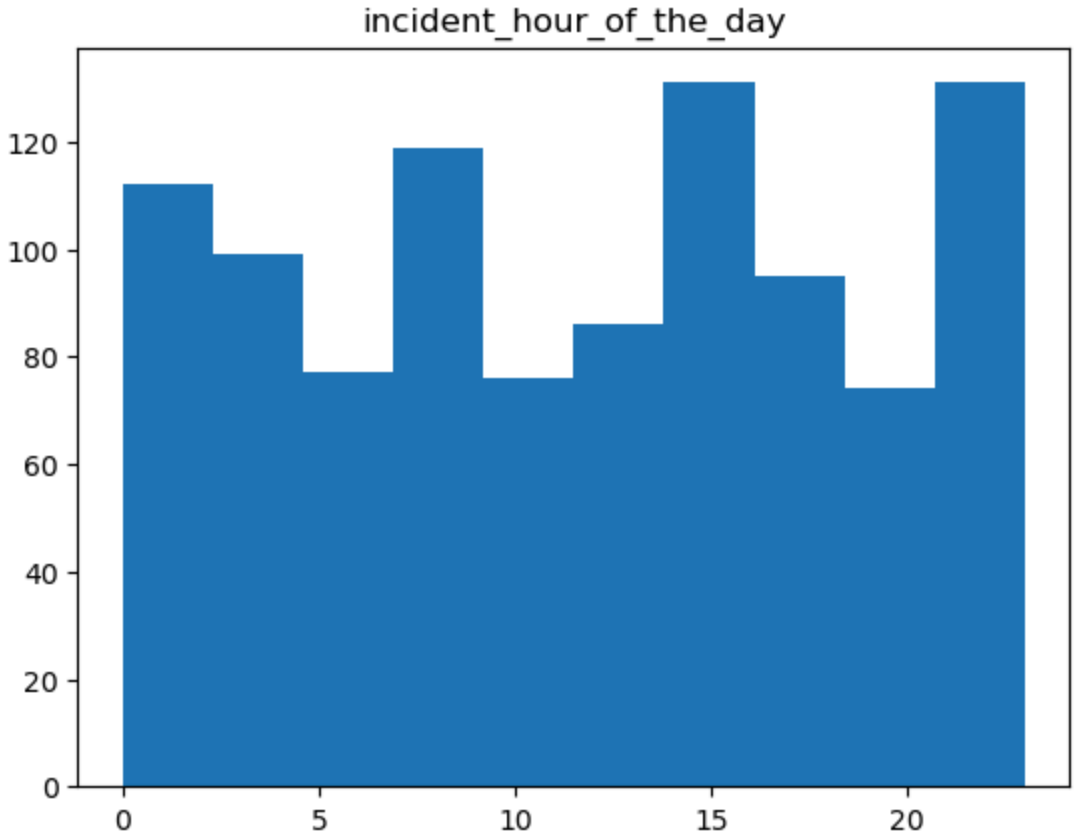

incident_severity
350
300
250
200
150
100
50
0
Major Damage
Minor Damage
Total Loss
Trivial Damage

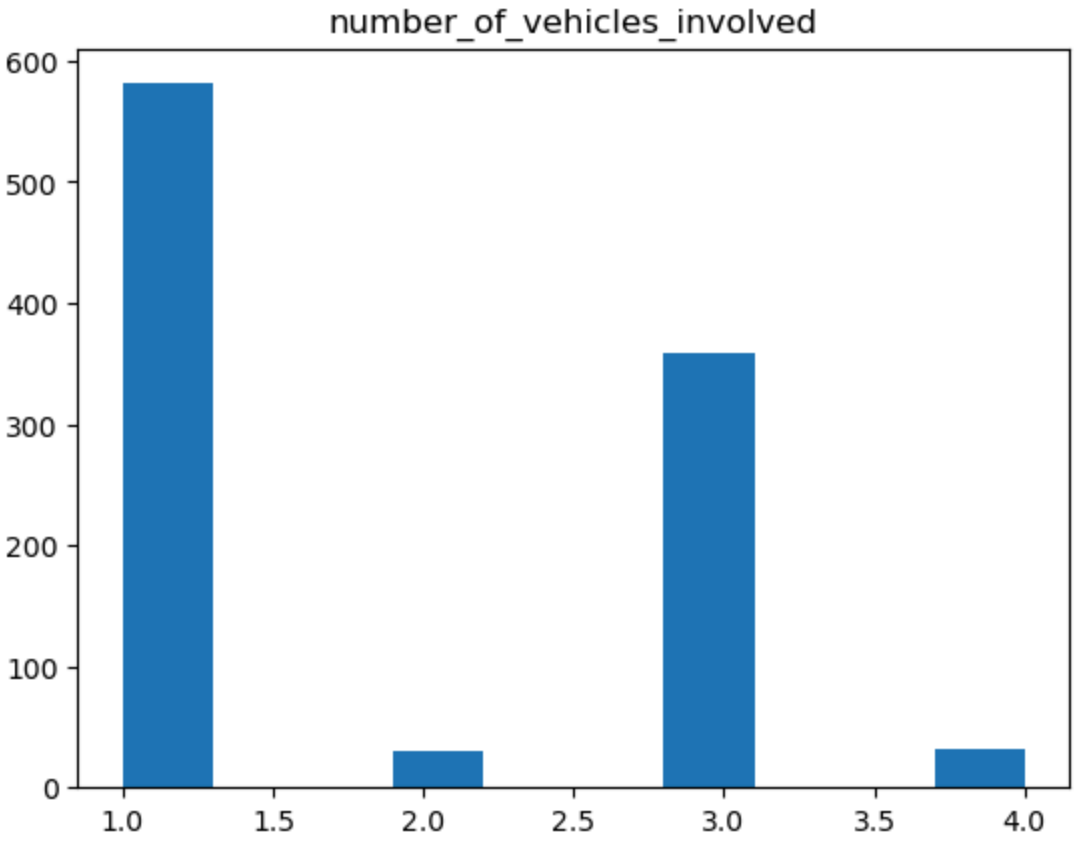

authorities_contacted
300
250
200
150
100
50
0
Police
None
Fire
Other
Ambulance

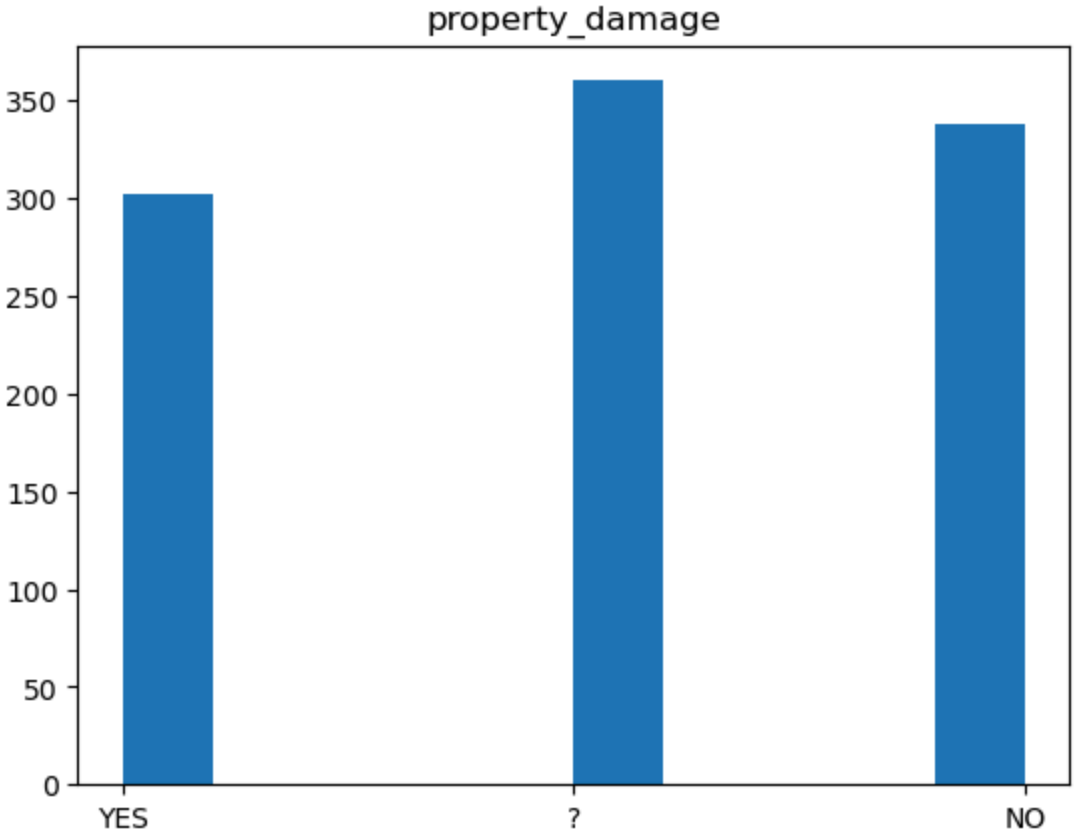

incident_state
250
200
150
100
50
0
SC
VA
NY
OH
WV
NC
PA

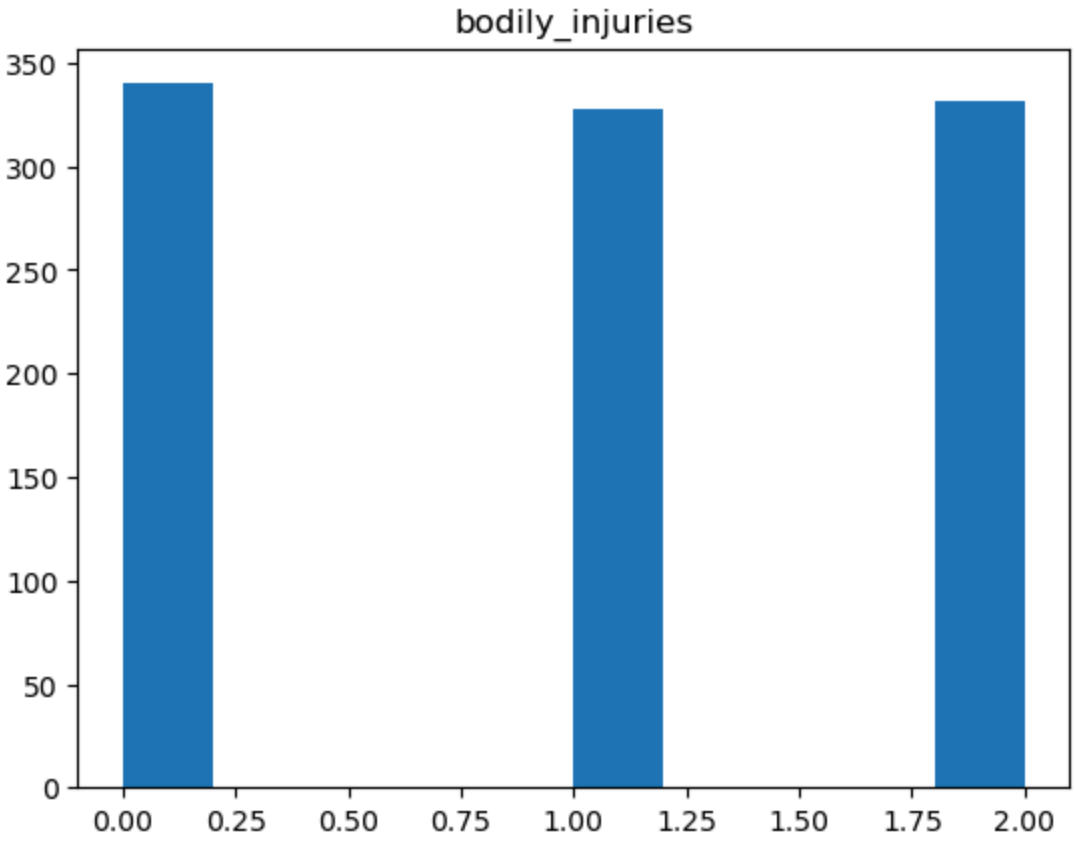

incident_city
160
140
120
100
80
60
40
20
0
Columbus
Riverwood
Arlington
Springfield
Hillsdale
Northbend
Northbrook

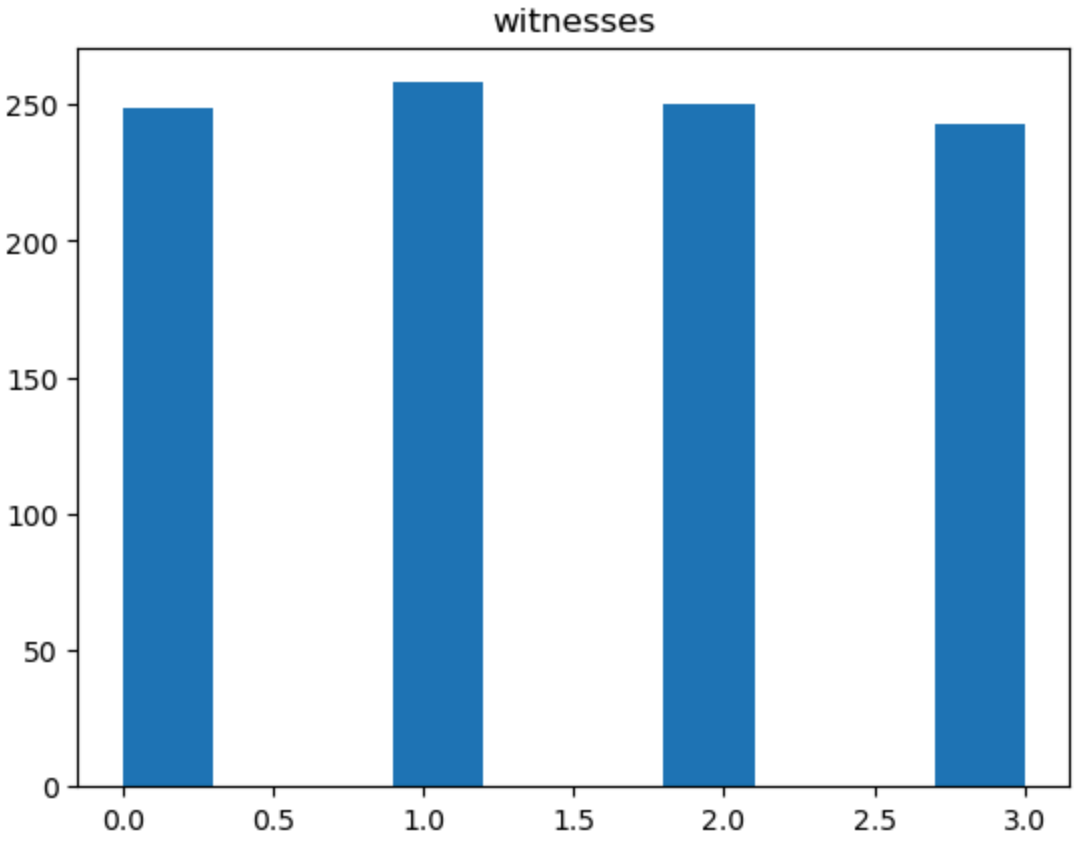

incident_location
100
80
60
40
20
0

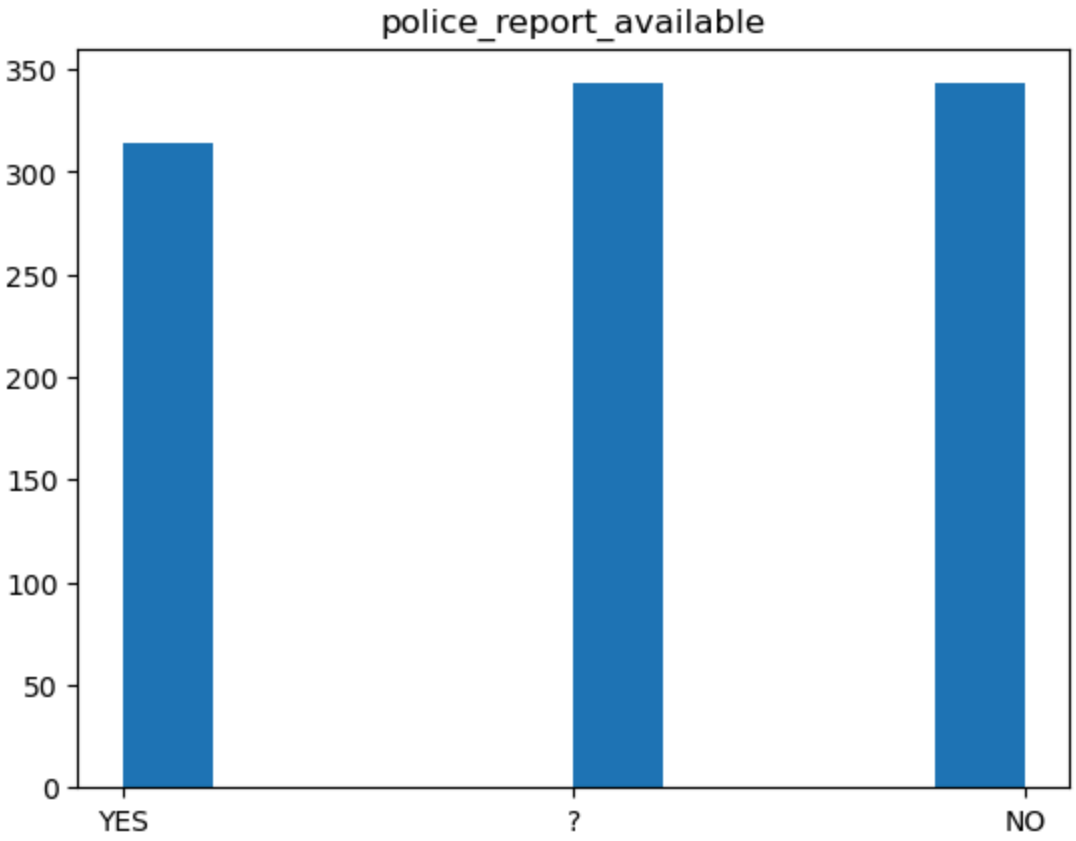

incident_hour_of_the_day
120
100
80
60
40
20
0
0
5
10
15
20

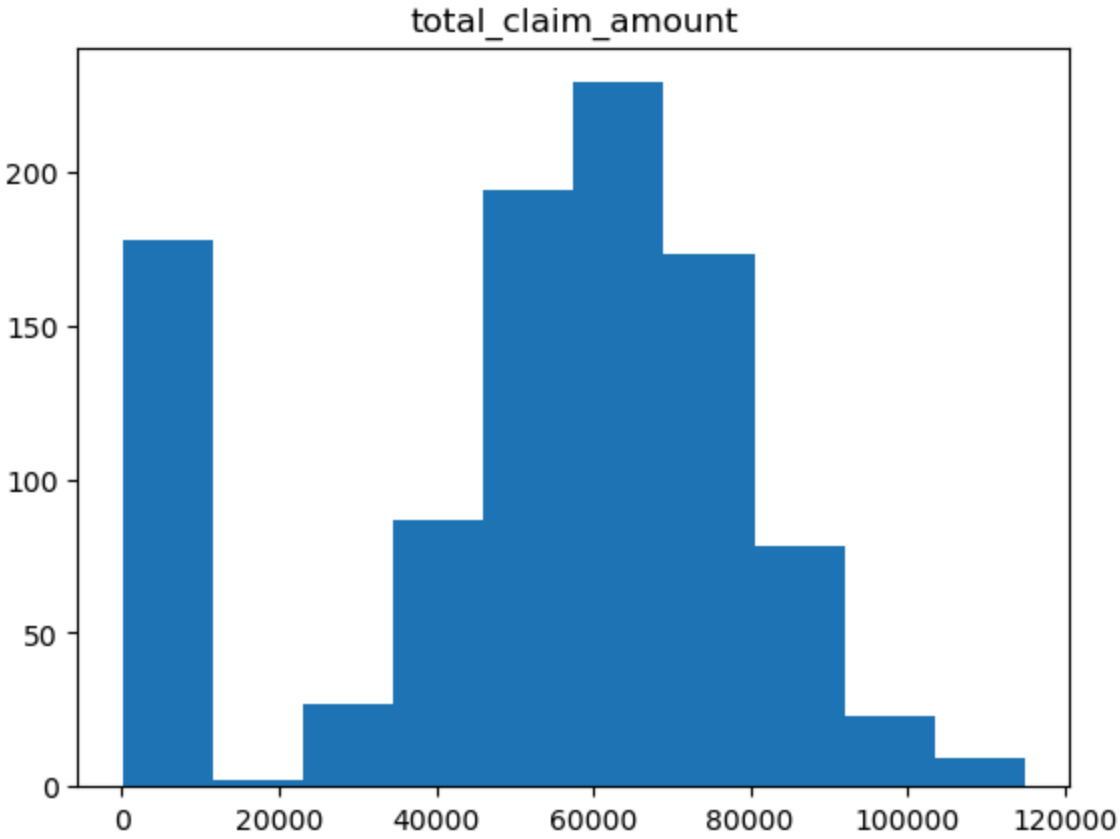

number_of_vehicles_involved
600
500
400
300
200
100
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0

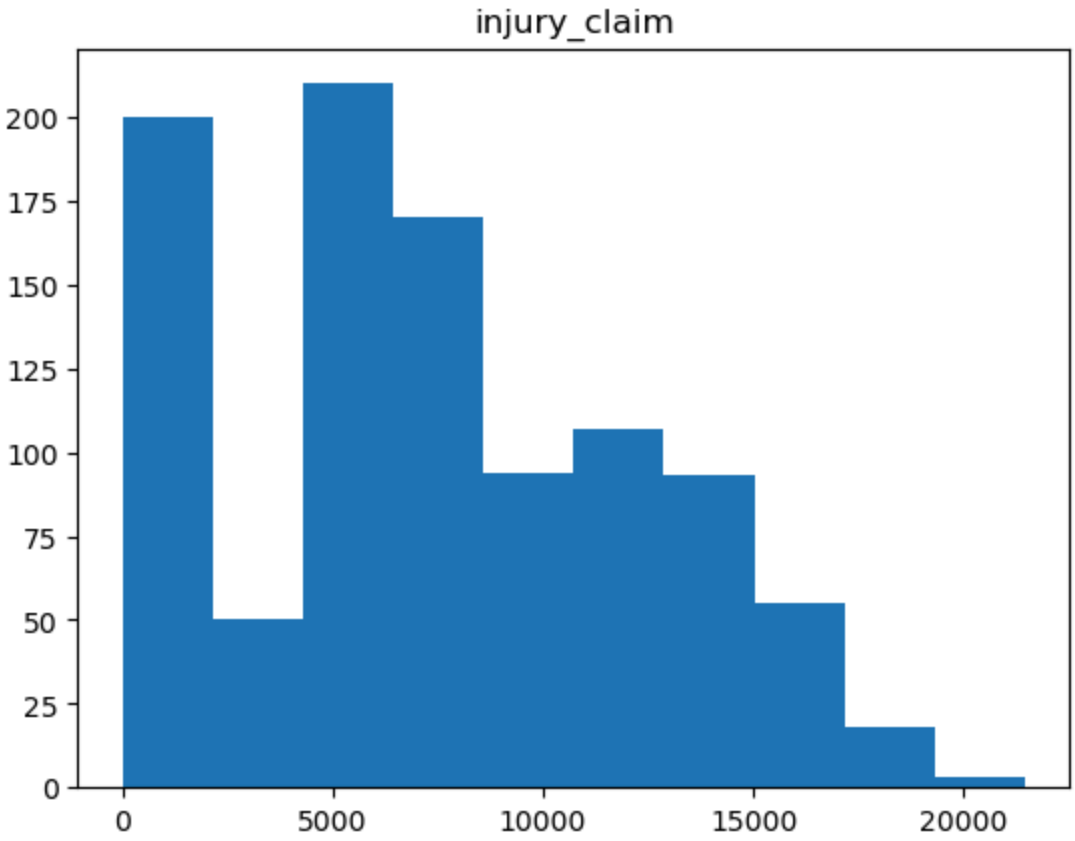

property_damage
350
300
250
200
150
100
50
0
YES
?
NO

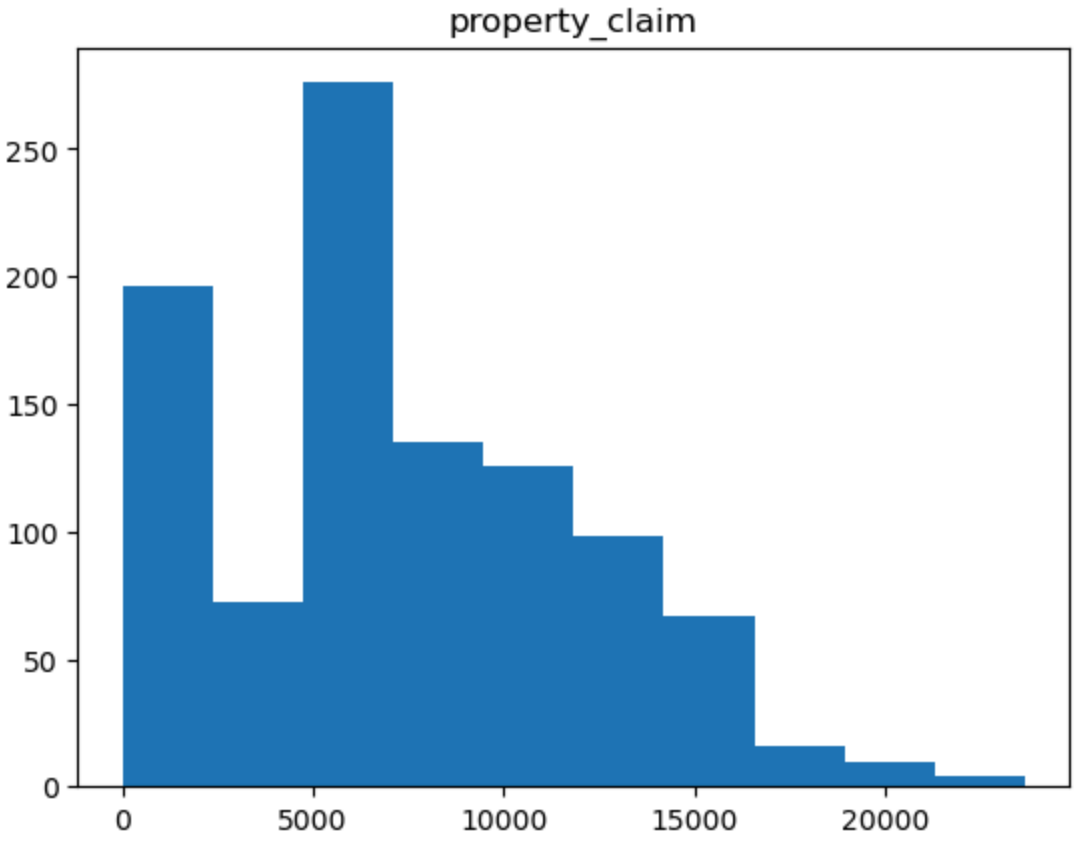

bodily_injuries
350
300
250
200
150
100
50
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

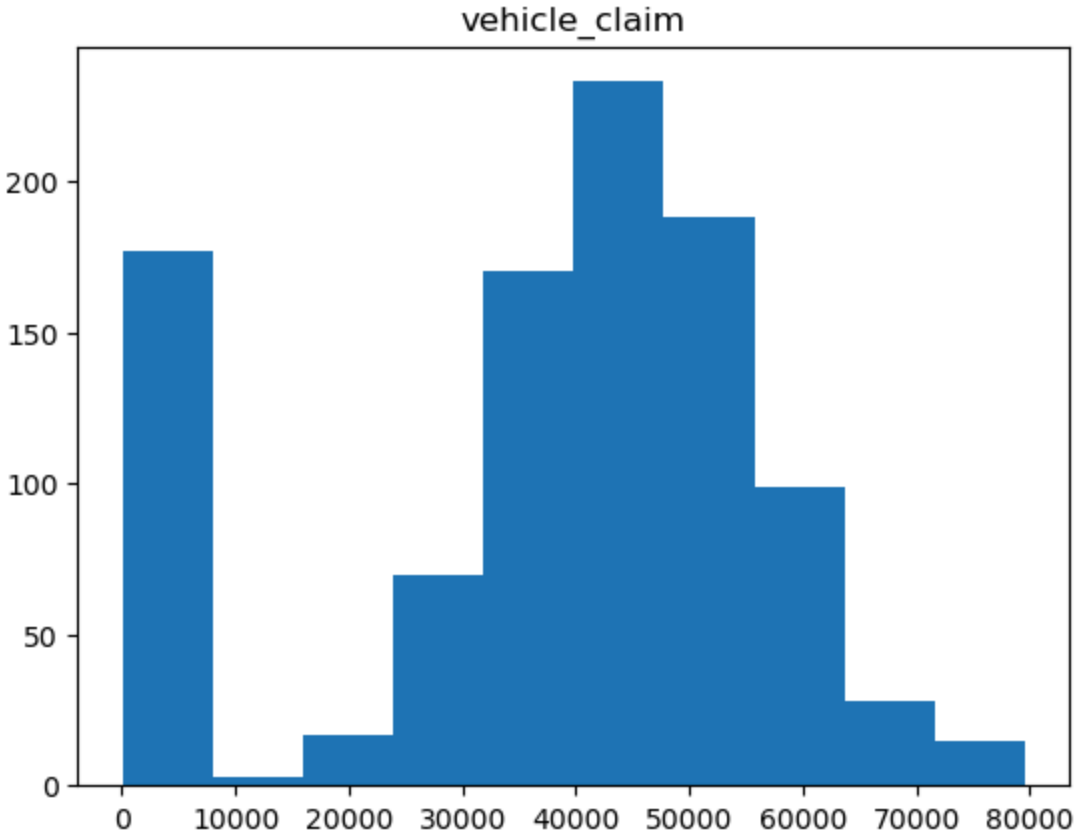

witnesses
250
200
150
100
50
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

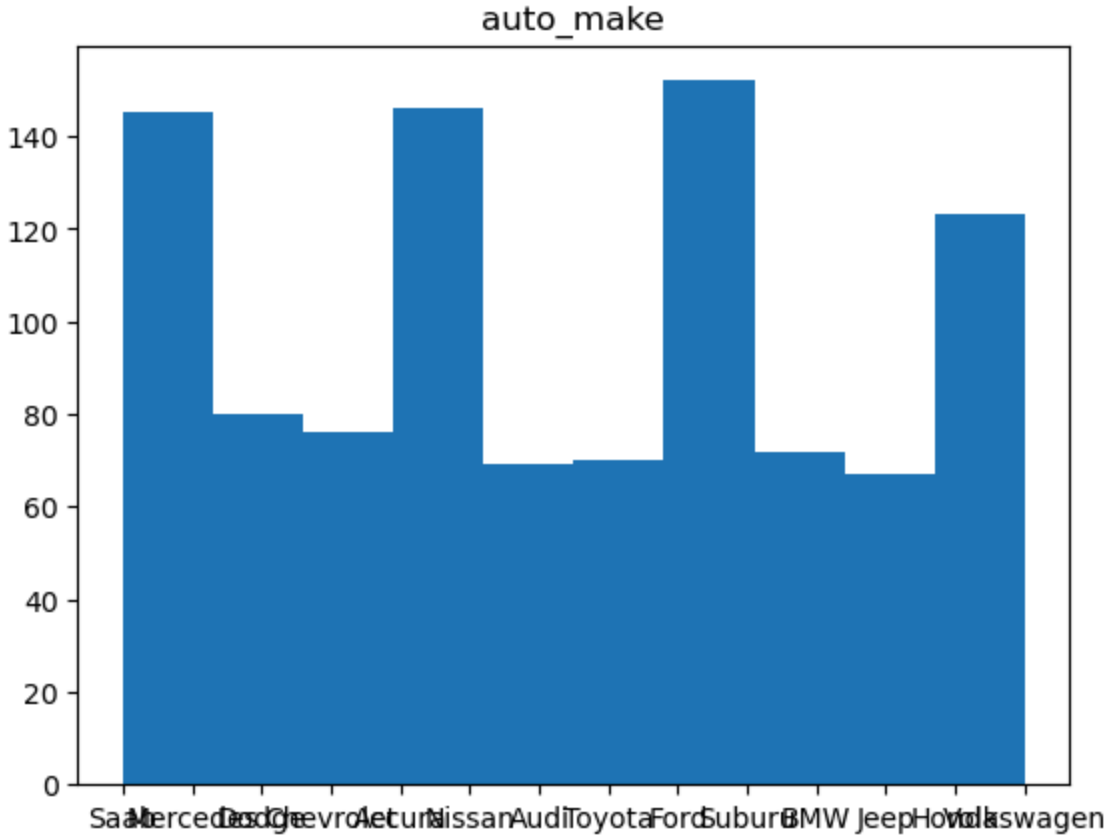

police_report_available
350
300
250
200
150
100
50
0
YES
?
NO

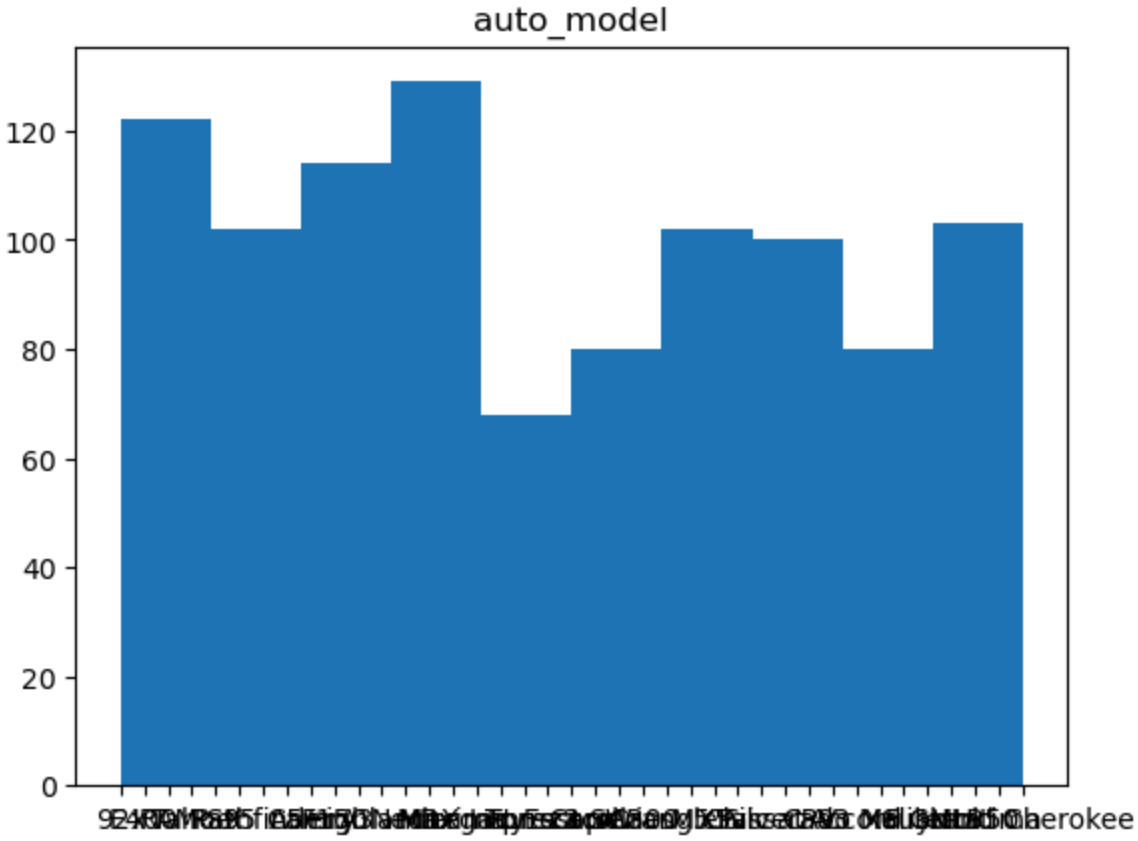

total_claim_amount
200
150
100
50
0
0
20000
40000
60000
80000
100000
120000

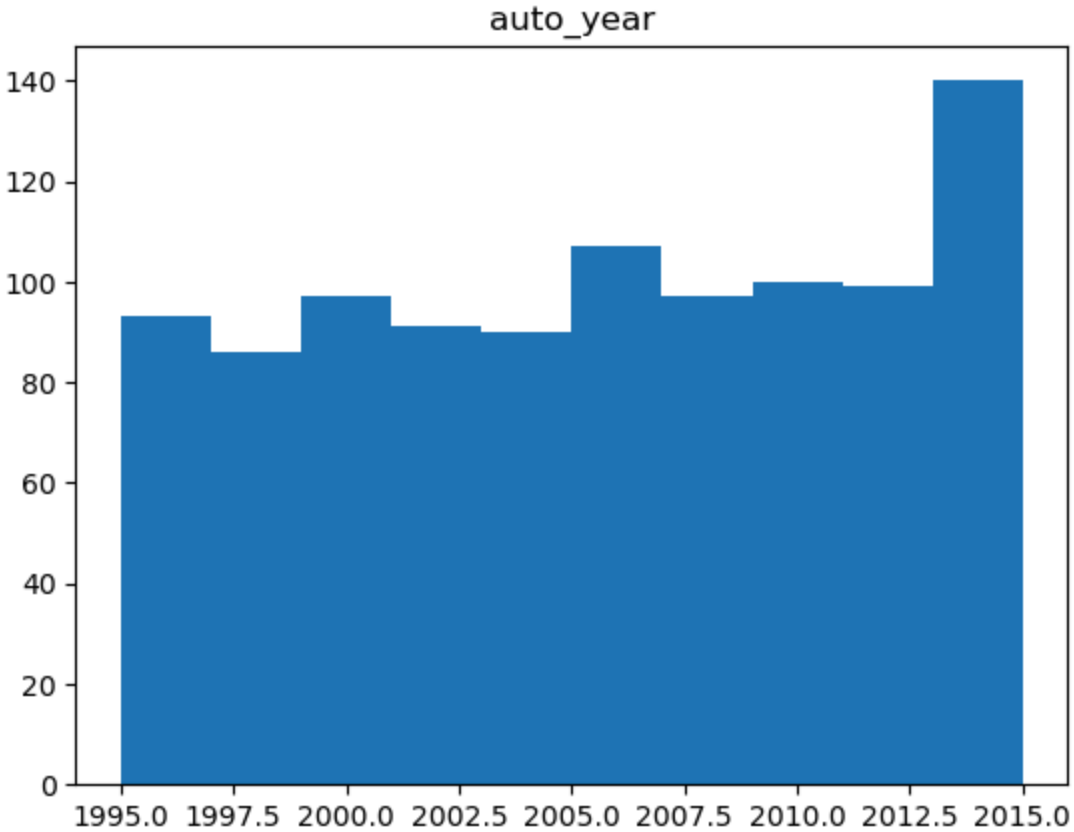

injury_claim
200
175
150
125
100
75
50
25
0
0
5000
10000
15000
20000

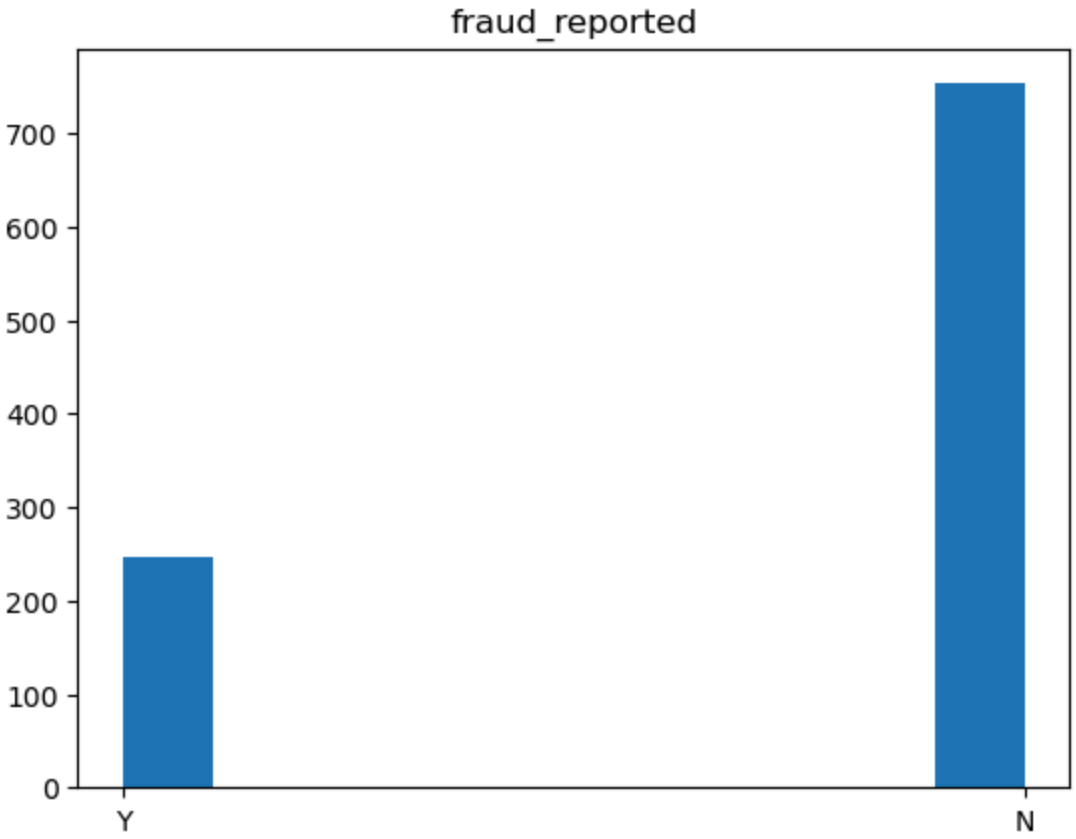

property_claim
250
200
150
100
50
0
0
5000
10000
15000
20000


vehicle_claim
200
150
100
50
0
0
10000
20000
30000
40000
50000
60000
70000
80000


auto_make
140
120
100
80
60
40
20
0
Saab
Mercedes
Dodge
Chevrolet
Acura
Nissan
Audi
Toyota
Ford
Suburu
BMW
Jeep
Honda
Volkswagen


auto_model
120
100
80
60
40
20
0
Cherokee


auto_year
140
120
100
80
60
40
20
0
1995.0
1997.5
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0



In [11]:
```
#Correlations
df.corr()
```

```
C:\Users\carol\AppData\Local\Temp\ipykernel_26688\1874986527.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.corr()
```

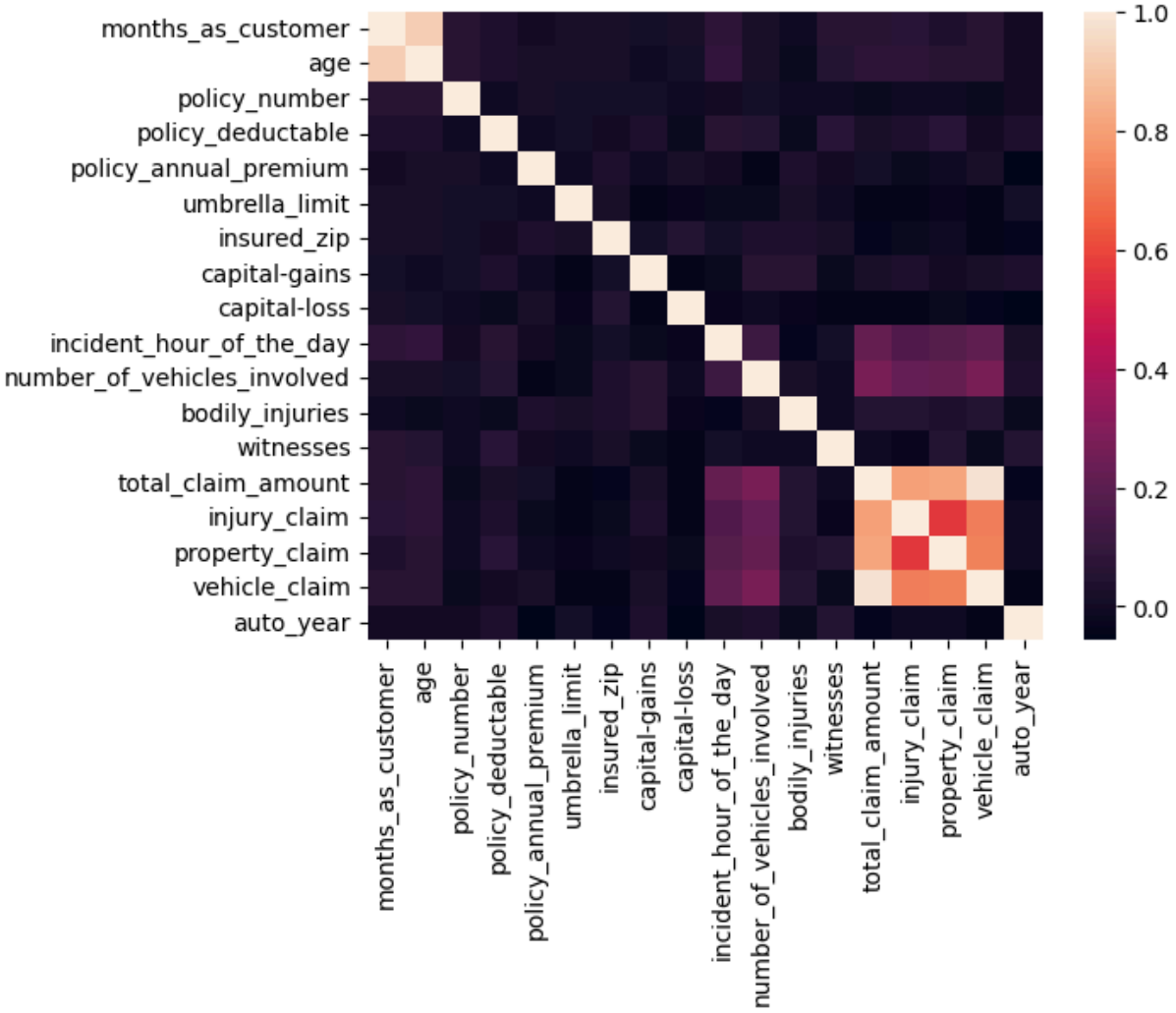
Out[11]:

	months_as_customer	age	policy_number	policy_deducta
months_as_customer	1.000000	0.922098	0.057555	0.026
age	0.922098	1.000000	0.059413	0.029
policy_number	0.057555	0.059413	1.000000	-0.006
policy_deductable	0.026807	0.029188	-0.006738	1.000
policy_annual_premium	0.005018	0.014404	0.022566	-0.003
umbrella_limit	0.015498	0.018126	0.008968	0.010
insured_zip	0.017895	0.025604	0.007083	0.004
capital-gains	0.006399	-0.007075	0.009802	0.035
capital-loss	0.020209	0.007368	-0.005669	-0.023
incident_hour_of_the_day	0.070639	0.087161	0.000113	0.060
number_of_vehicles_involved	0.014736	0.022102	0.013432	0.051
bodily_injuries	-0.010162	-0.015679	-0.004558	-0.022
witnesses	0.058383	0.052359	-0.012661	0.066
total_claim_amount	0.062108	0.069863	-0.018009	0.022
injury_claim	0.065329	0.075522	-0.008762	0.039
property_claim	0.034940	0.060898	-0.010678	0.064
vehicle_claim	0.061013	0.062588	-0.020184	0.005
auto_year	-0.000292	0.001354	-0.000183	0.026

In [12]:
```
corr=df.corr()
sns.heatmap(corr)
```

```
C:\Users\carol\AppData\Local\Temp\ipykernel_26688\133148687.py:1: FutureWarning: The
default value of numeric_only in DataFrame.corr is deprecated. In a future version,
it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  corr=df.corr()
```

Out[12]: <Axes: >

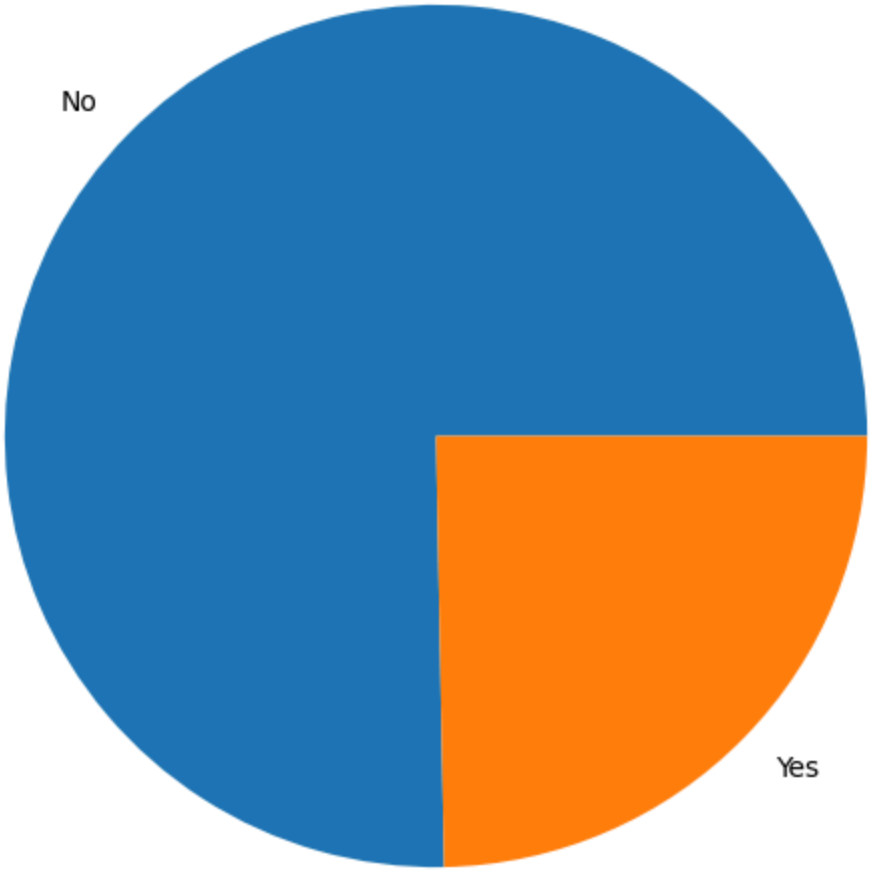


In [13]:
```
#Investigate the distribution of the fraud reported column
num_fraud = df["fraud_reported"].value_counts()
num_fraud
```

Out[13]:
```
N    753
Y    247
Name: fraud_reported, dtype: int64
```

In [14]:
```
fig = plt.figure(figsize=(10, 7))
plt.pie(num_fraud, labels=["No", "Yes"])
```

Out[14]:
```
([<matplotlib.patches.Wedge at 0x261a0d61940>,
  <matplotlib.patches.Wedge at 0x261a0d61100>],
 [Text(-0.7851136132870644, 0.7704522141128092, 'No'),
  Text(0.785113649354535, -0.7704521773589873, 'Yes')])
```



In [ ]:

In [15]:
```
#DATA CLEANING
```

In [16]:
```
#change capital loss to positive (+) intergers for clarity
df["capital-loss"] = df["capital-loss"]*-1
```

In [17]:
```
#Quantifying features
df['fraud_reported'] = df['fraud_reported'].map({'Y': 1, 'N': 0})
df['insured_sex'] = df['insured_sex'].map({'FEMALE': 1, 'MALE': 0})
df['insured_education_level'] = df['insured_education_level'].map({'High School': 0
df['incident_severity'] = df['incident_severity'].map({'Trivial Damage': 0, 'Minor
```

In [ ]:

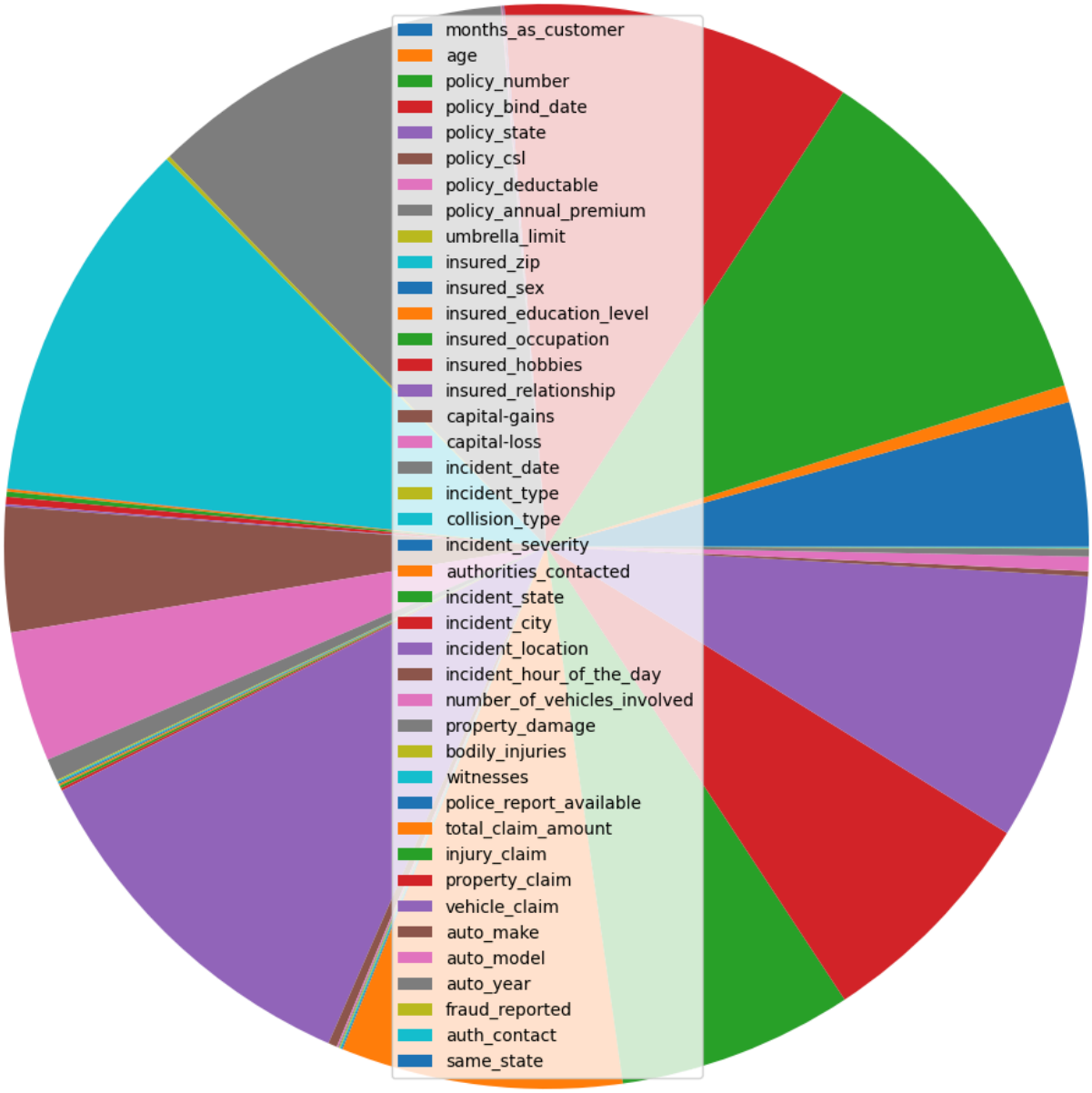
In [18]:
```
#Adding Columns
df['auth_contact'] = np.where(df['authorities_contacted']== "None", 0, 1) #designat
df['same_state'] = np.where(df['policy_state']== df["incident_state"], 1, 0) #wheth
```

In [19]:
```
#Look at how many unique values in the features
num_unique_values = df.nunique()
column_names=list(df)
```

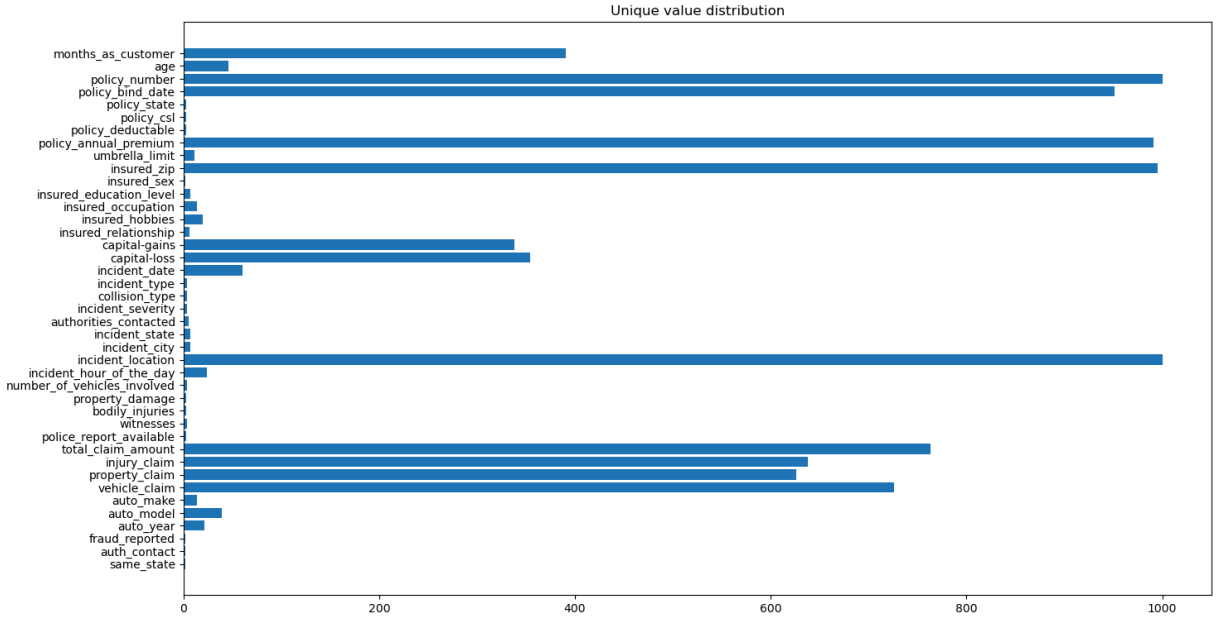
In [20]:
```
num_unique_values
```

Out[20]:
```
months_as_customer             391
age                             46
policy_number                 1000
policy_bind_date               951
policy_state                     3
policy_csl                       3
policy_deductable                3
policy_annual_premium          991
umbrella_limit                  11
insured_zip                    995
insured_sex                      2
insured_education_level          7
insured_occupation              14
insured_hobbies                 20
insured_relationship             6
capital-gains                  338
capital-loss                   354
incident_date                   60
incident_type                    4
collision_type                   4
incident_severity                4
authorities_contacted            5
incident_state                   7
incident_city                    7
incident_location             1000
incident_hour_of_the_day        24
number_of_vehicles_involved      4
property_damage                  3
bodily_injuries                  3
witnesses                        4
police_report_available          3
total_claim_amount             763
injury_claim                   638
property_claim                 626
vehicle_claim                  726
auto_make                       14
auto_model                      39
auto_year                       21
fraud_reported                   2
auth_contact                     2
same_state                       2
dtype: int64
```

In [21]:
```
plt.pie(num_unique_values, radius=3)
plt.legend(labels=column_names, loc="center")
plt.show()
```



In [22]:
```
fig, ax = plt.subplots(figsize =(16, 9))
ax.barh(column_names, num_unique_values)
ax.invert_yaxis()
ax.set_title('Unique value distribution', loc ='center', )

plt.show()
```



In [23]:
```
#There are certain features which contain "?" as a value.

#collision type has "?"
#property damage has "?"

df[df == "?"].count()
```

Out[23]:
```
months_as_customer              0
age                             0
policy_number                   0
policy_bind_date                0
policy_state                    0
policy_csl                      0
policy_deductable               0
policy_annual_premium           0
umbrella_limit                  0
insured_zip                     0
insured_sex                     0
insured_education_level         0
insured_occupation              0
insured_hobbies                 0
insured_relationship            0
capital-gains                   0
capital-loss                    0
incident_date                   0
incident_type                   0
collision_type                178
incident_severity               0
authorities_contacted           0
incident_state                  0
incident_city                   0
incident_location               0
incident_hour_of_the_day        0
number_of_vehicles_involved     0
property_damage               360
bodily_injuries                 0
witnesses                       0
police_report_available       343
total_claim_amount              0
injury_claim                    0
property_claim                  0
vehicle_claim                   0
auto_make                       0
auto_model                      0
auto_year                       0
fraud_reported                  0
auth_contact                    0
same_state                      0
dtype: int64
```

In [24]:
```
list(df)
```

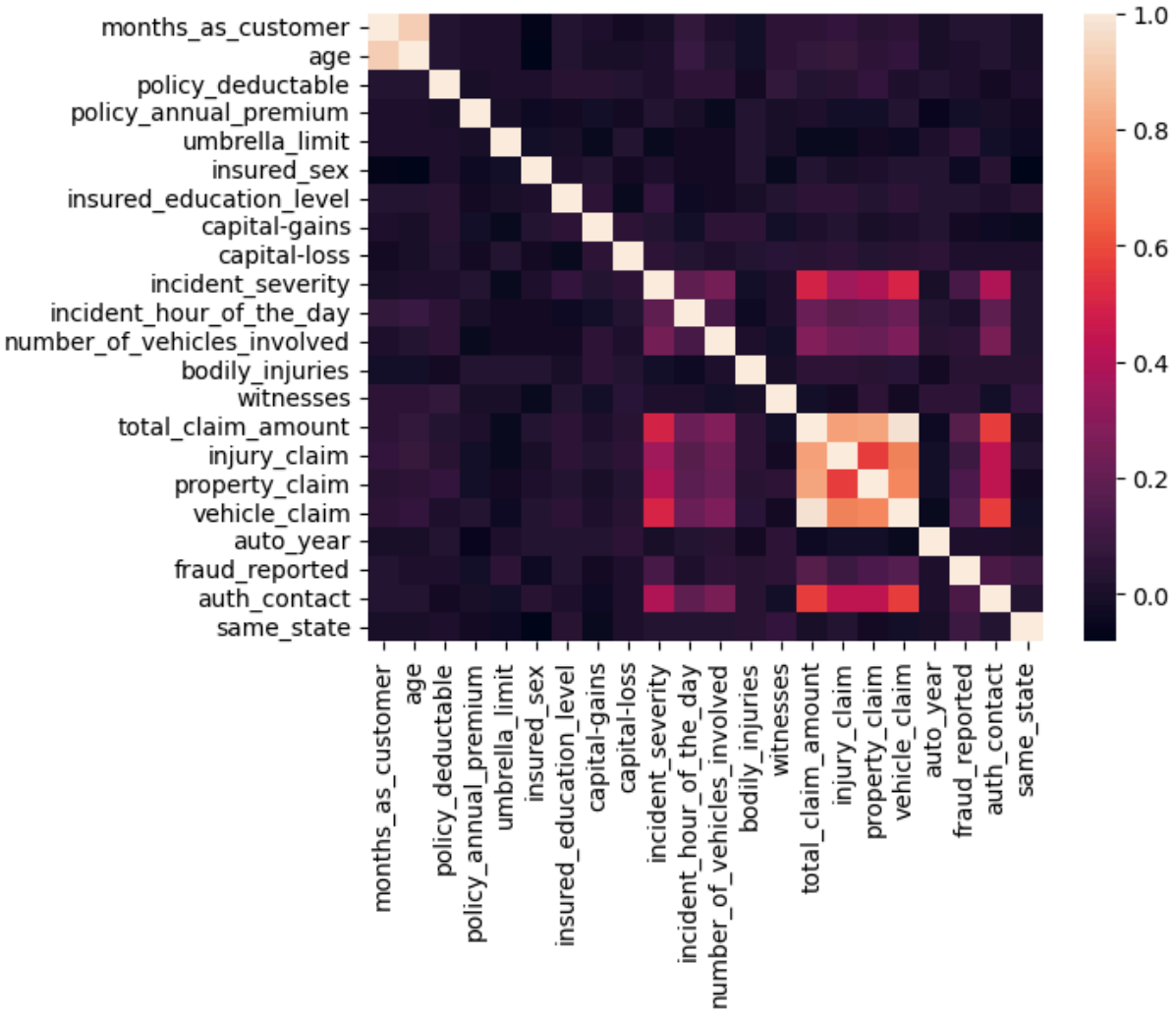
Out[24]:
```
['months_as_customer',
 'age',
 'policy_number',
 'policy_bind_date',
 'policy_state',
 'policy_csl',
 'policy_deductable',
 'policy_annual_premium',
 'umbrella_limit',
 'insured_zip',
 'insured_sex',
 'insured_education_level',
 'insured_occupation',
 'insured_hobbies',
 'insured_relationship',
 'capital-gains',
 'capital-loss',
 'incident_date',
 'incident_type',
 'collision_type',
 'incident_severity',
 'authorities_contacted',
 'incident_state',
 'incident_city',
 'incident_location',
 'incident_hour_of_the_day',
 'number_of_vehicles_involved',
 'property_damage',
 'bodily_injuries',
 'witnesses',
 'police_report_available',
 'total_claim_amount',
 'injury_claim',
 'property_claim',
 'vehicle_claim',
 'auto_make',
 'auto_model',
 'auto_year',
 'fraud_reported',
 'auth_contact',
 'same_state']
```

In [25]:
```
#Dropping Columns
df = df.drop("policy_number", axis=1)
df = df.drop("incident_location", axis=1)
df = df.drop("incident_city", axis=1)
df = df.drop("incident_date", axis=1)
df = df.drop("auto_model", axis=1)
df = df.drop("policy_bind_date", axis=1)
df = df.drop("policy_csl", axis=1)
df = df.drop("policy_state", axis=1)
df = df.drop("incident_state", axis=1)
df = df.drop("insured_zip", axis=1) #highly unique
```

In [26]:
```
#Look at correlations after substantial changes
corr=df.corr()
sns.heatmap(corr)
```

```
C:\Users\carol\AppData\Local\Temp\ipykernel_26688\605927212.py:2: FutureWarning: The
default value of numeric_only in DataFrame.corr is deprecated. In a future version,
it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  corr=df.corr()
```

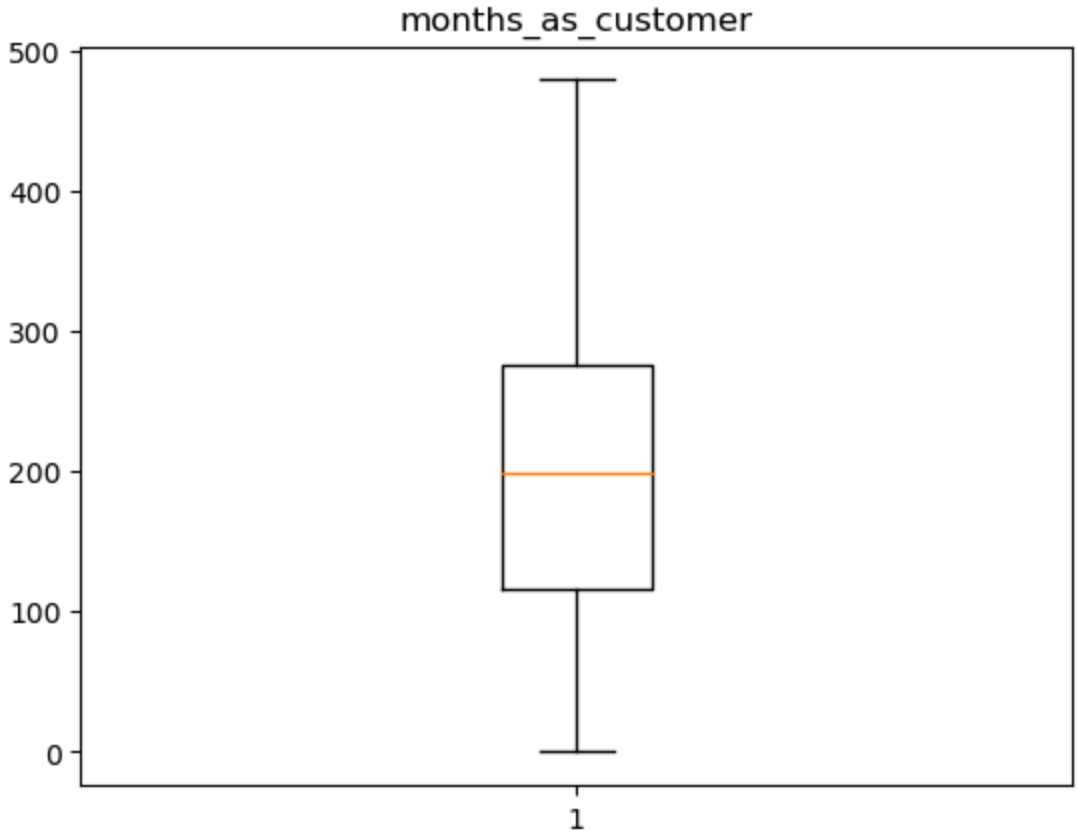
Out[26]: <Axes: >

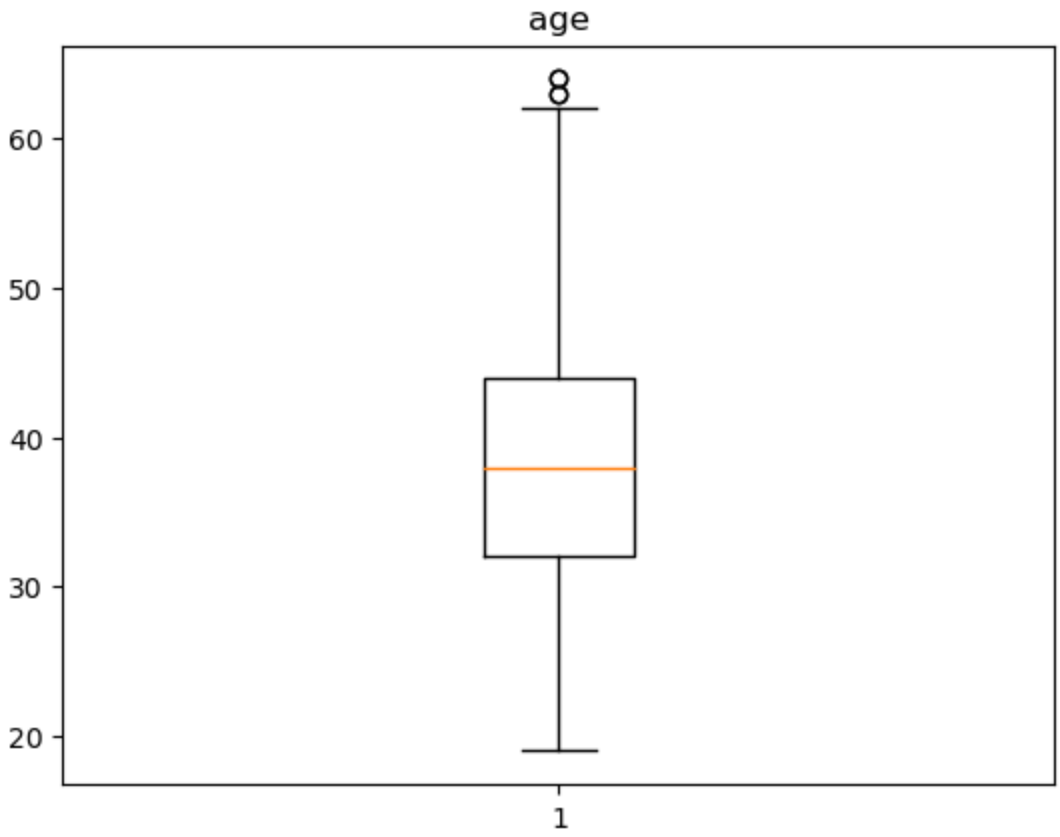


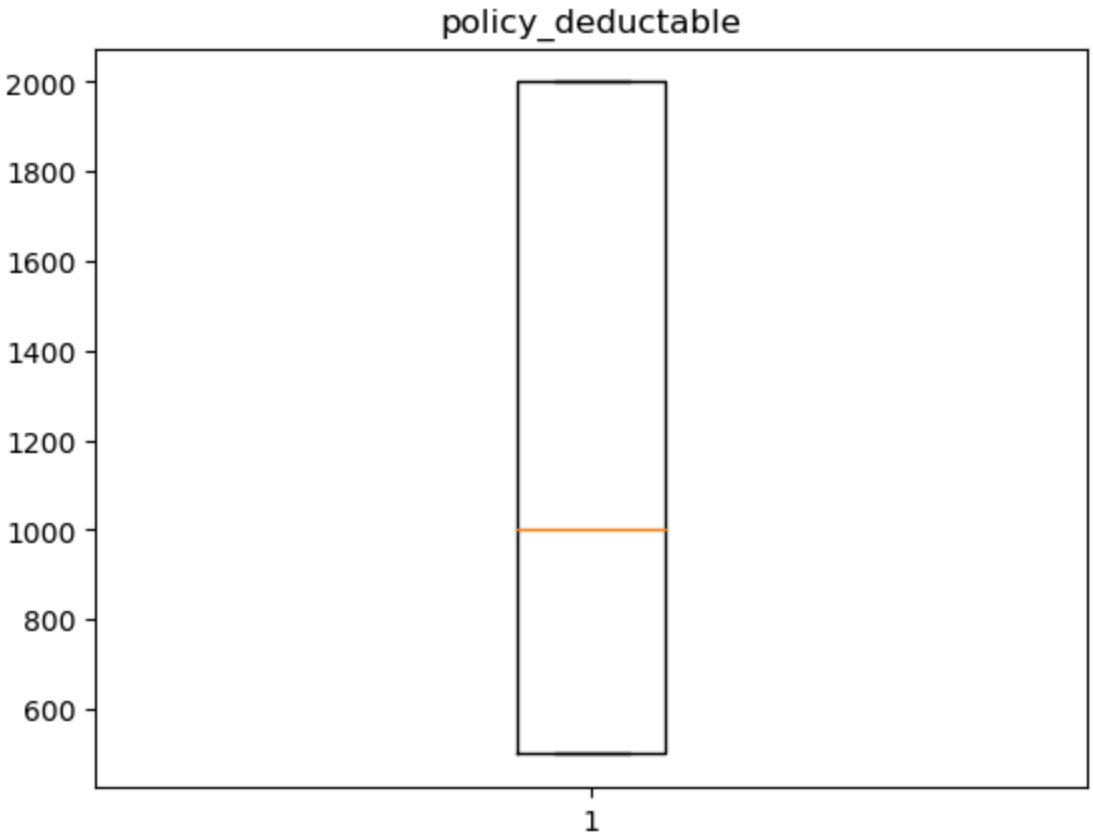
In [27]:
```
import matplotlib.pyplot as plt
```

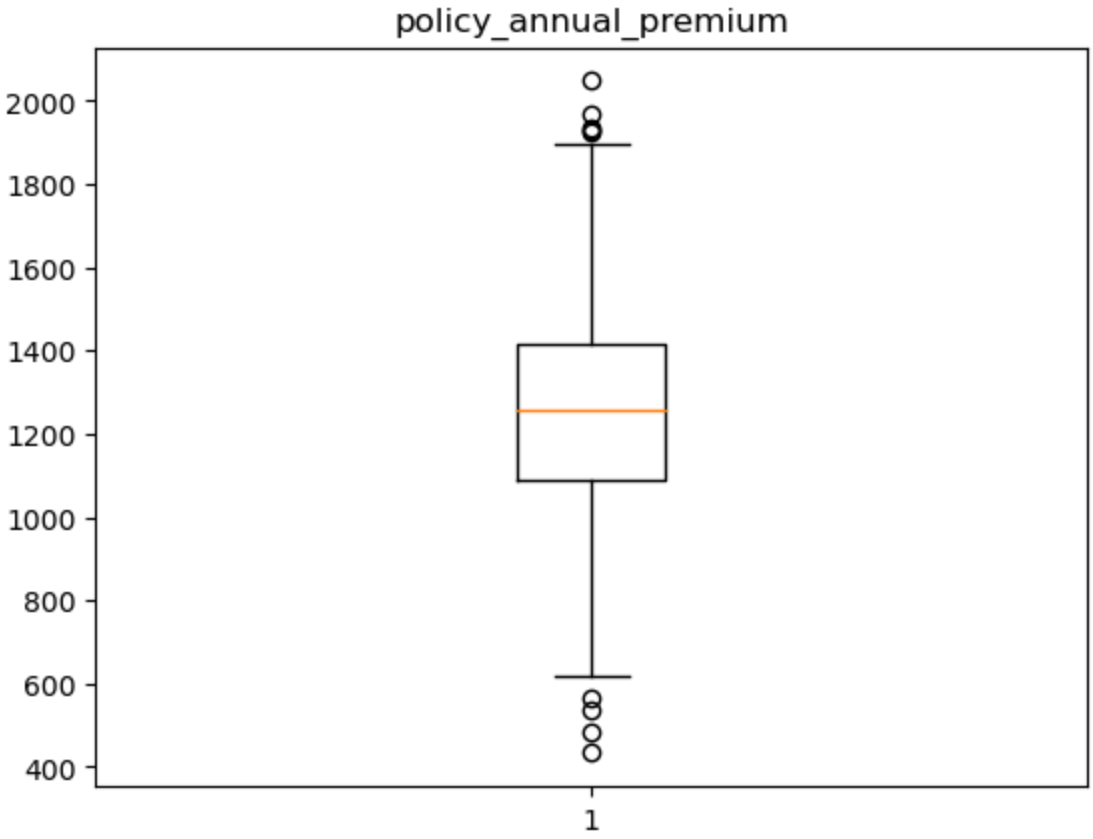
In [28]:
```
quant_cols=['months_as_customer','age','policy_deductable','policy_annual_premium',
 'capital-loss','incident_severity','incident_hour_of_the_day','number_of_vehicles_
```

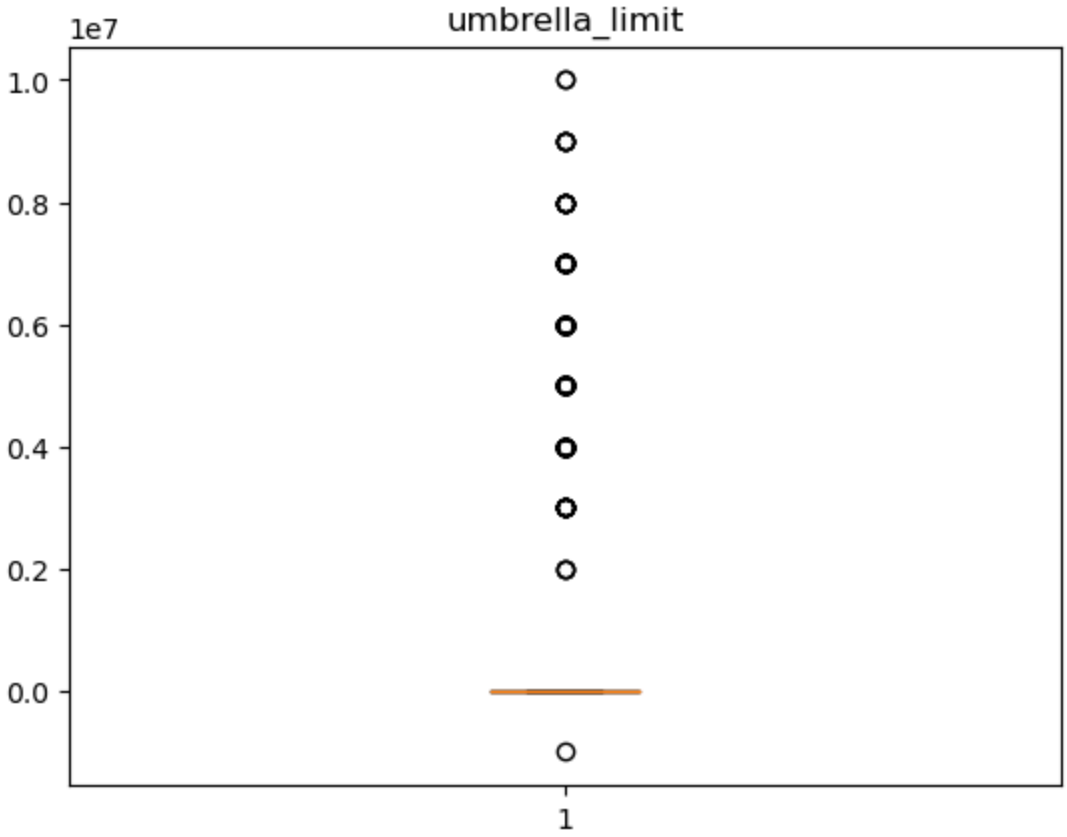
In [29]:
```
#Box Plots to look for interesting relationships and outliers.
for i in quant_cols:
    plt.boxplot(df[i])
    plt.title(i)
    plt.show()
```

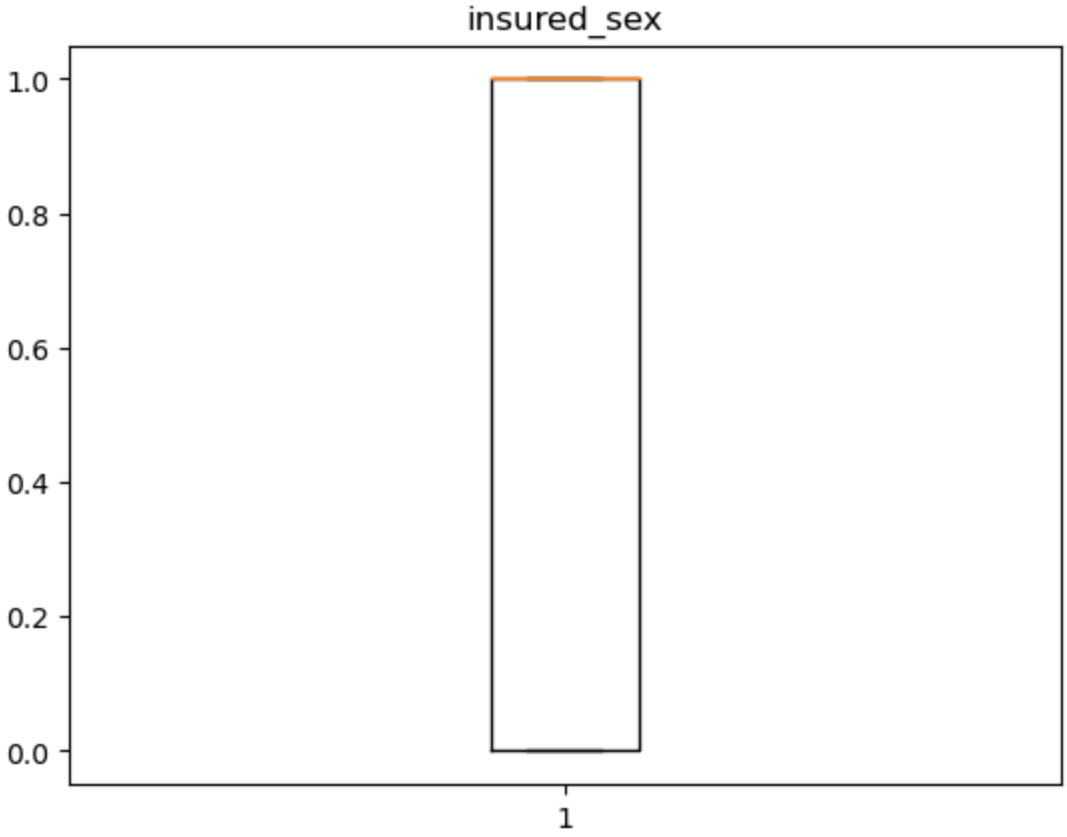

months_as_customer
500
400
300
200
100
0
1

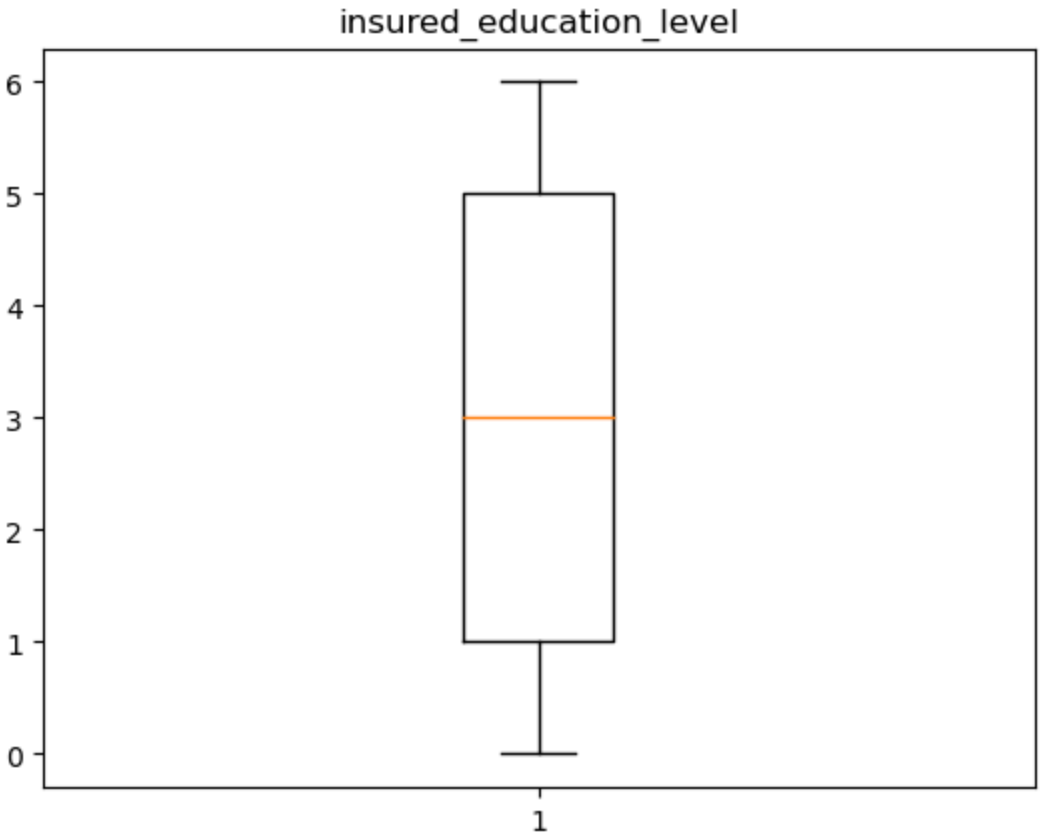

age
60
50
40
30
20
1

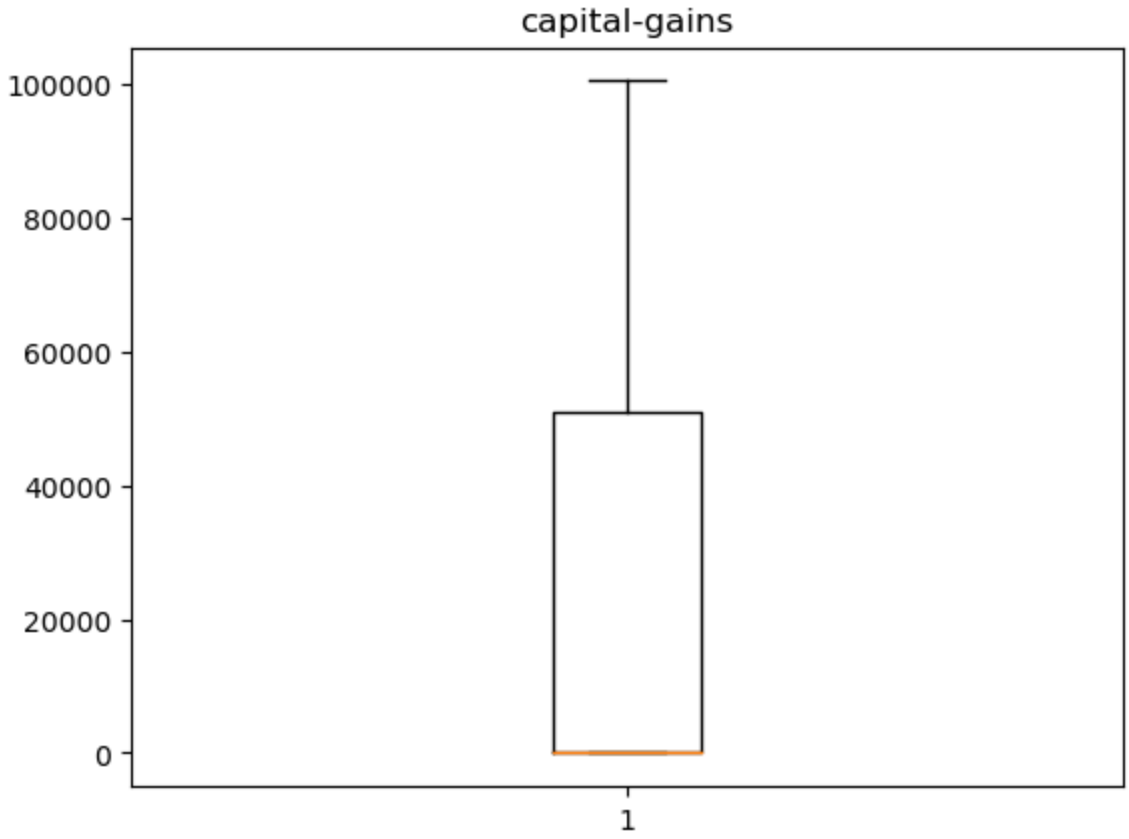

policy_deductable
2000
1800
1600
1400
1200
1000
800
600
1

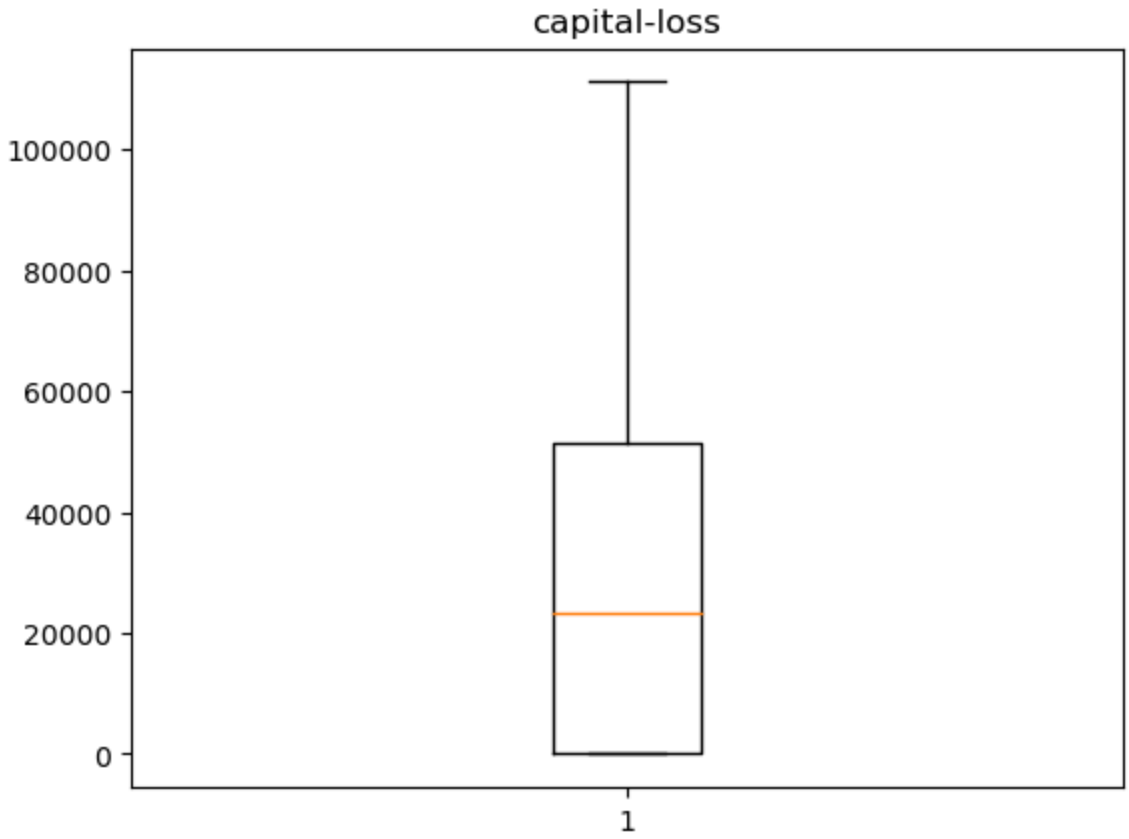

policy_annual_premium
2000
1800
1600
1400
1200
1000
800
600
400
1

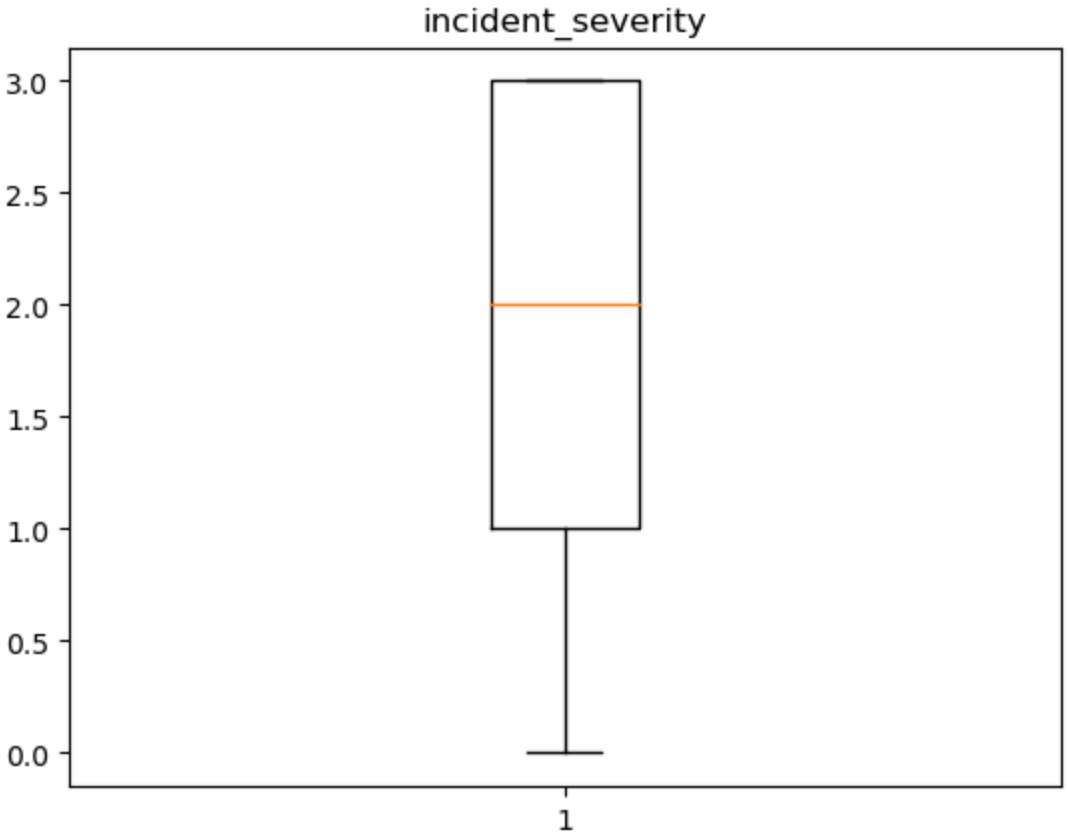

umbrella_limit
1e7
1.0
0.8
0.6
0.4
0.2
0.0
1

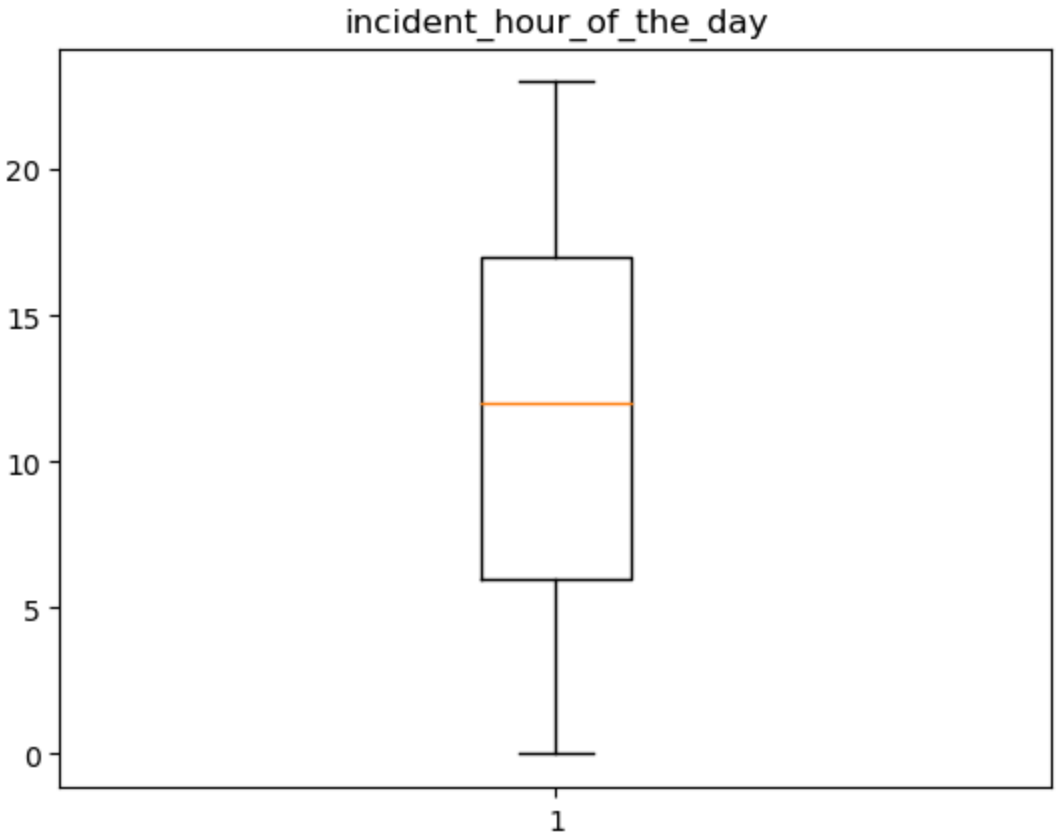

insured_sex
1.0
0.8
0.6
0.4
0.2
0.0
1

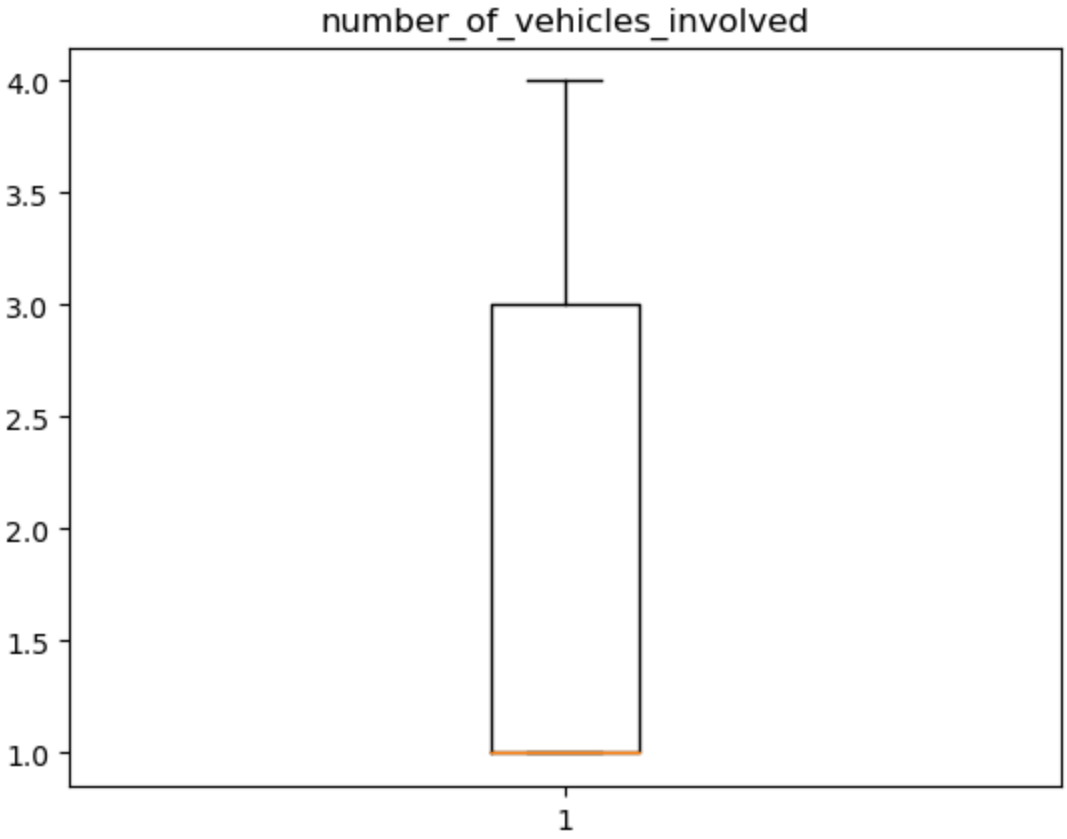

insured_education_level
6
5
4
3
2
1
0
1


capital-gains
100000
80000
60000
40000
20000
0
1

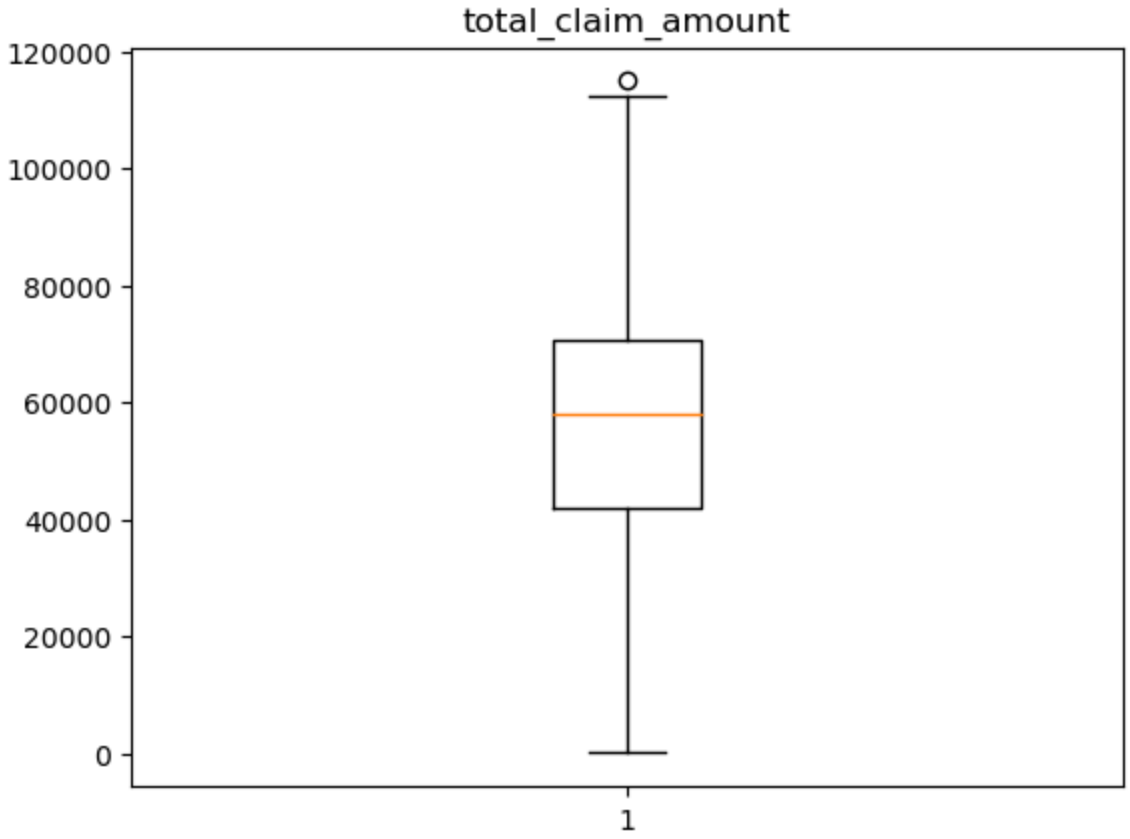

capital-loss
100000
80000
60000
40000
20000
0
1

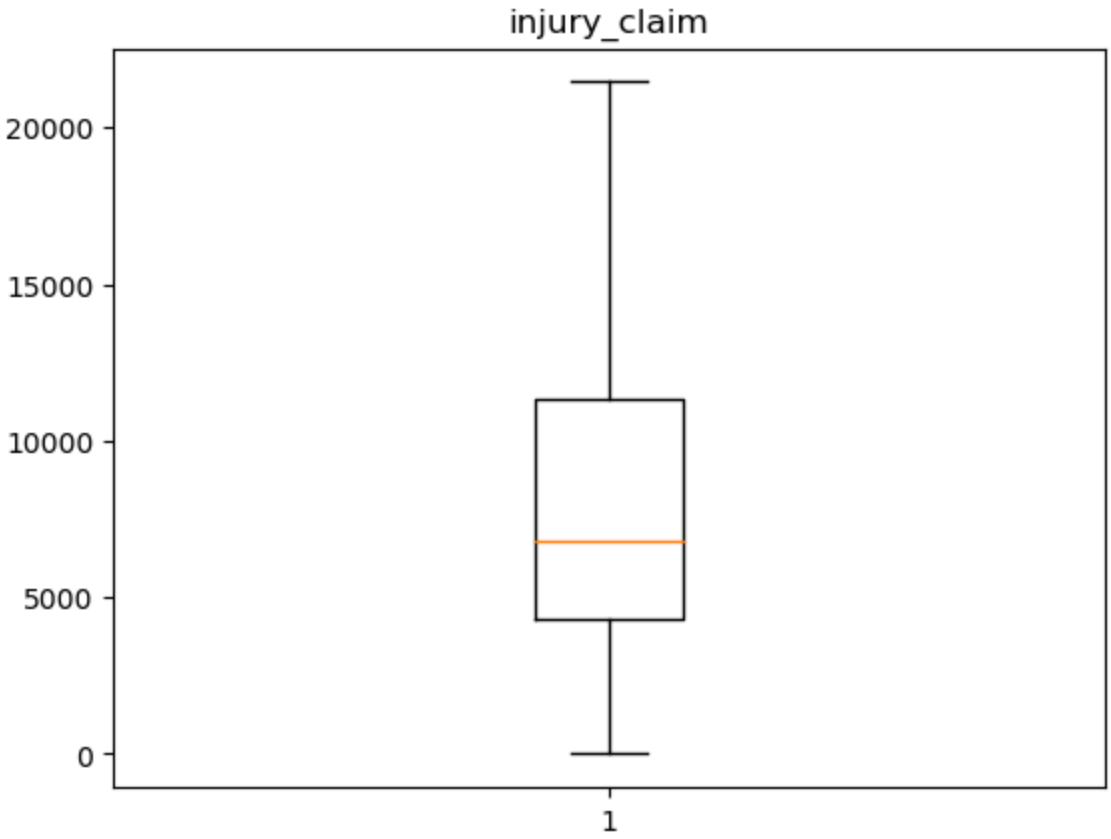

incident_severity
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1

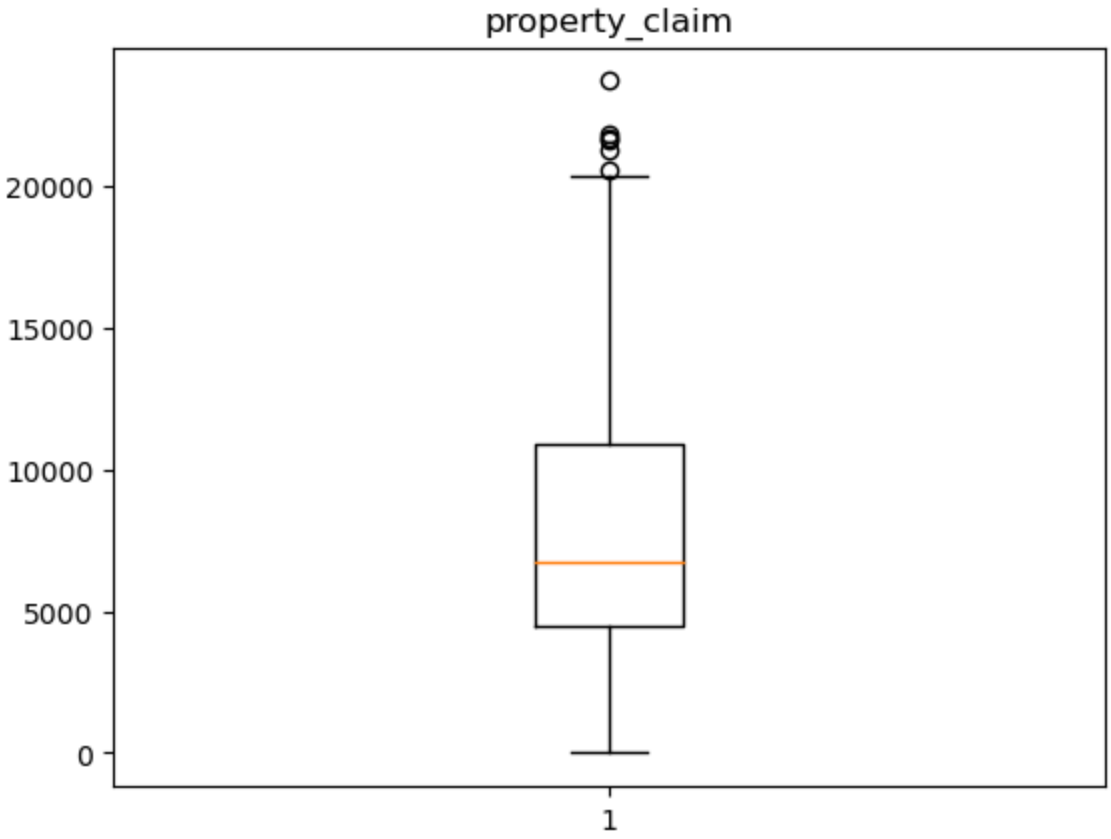

incident_hour_of_the_day
20
15
10
5
0
1

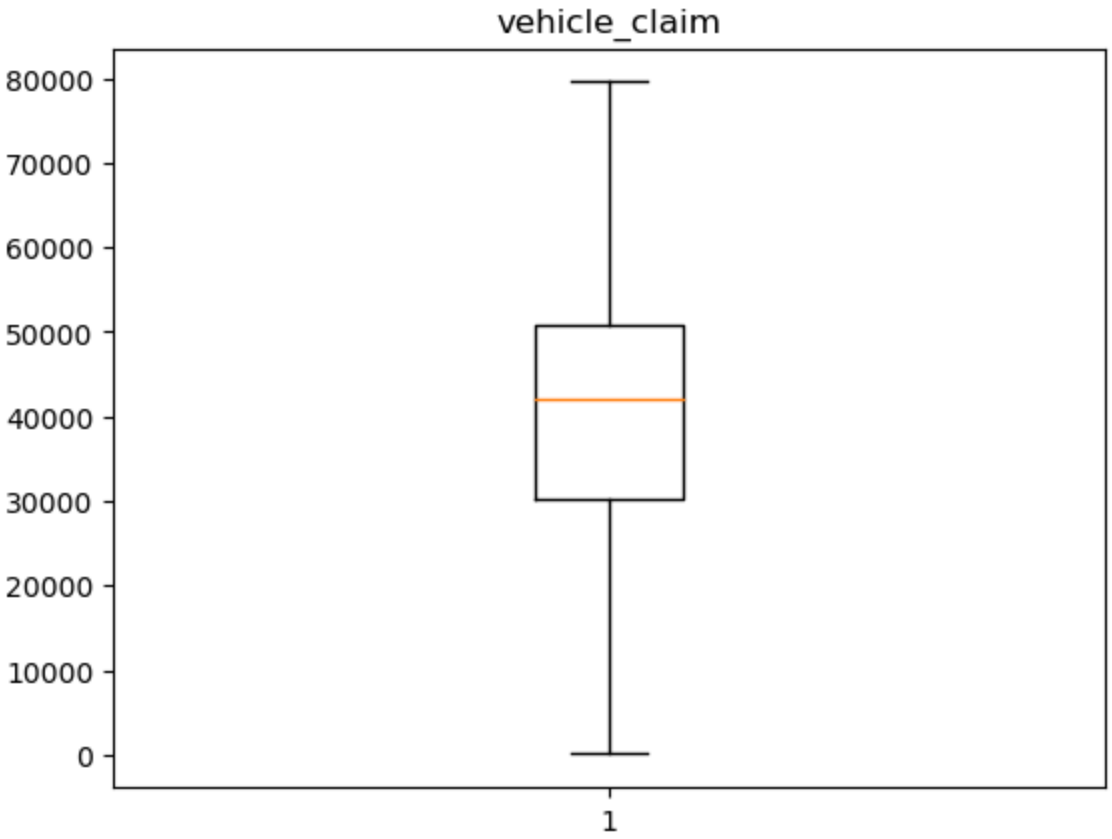

number_of_vehicles_involved
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1

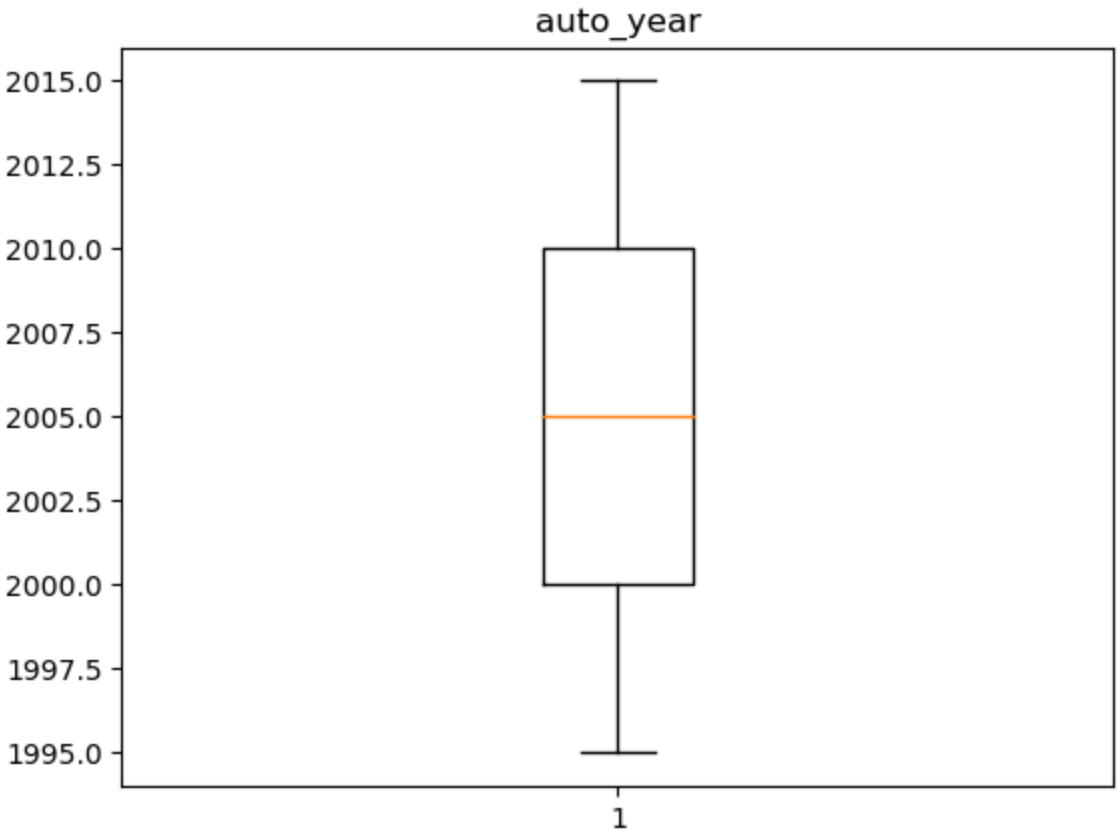

witnesses
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1

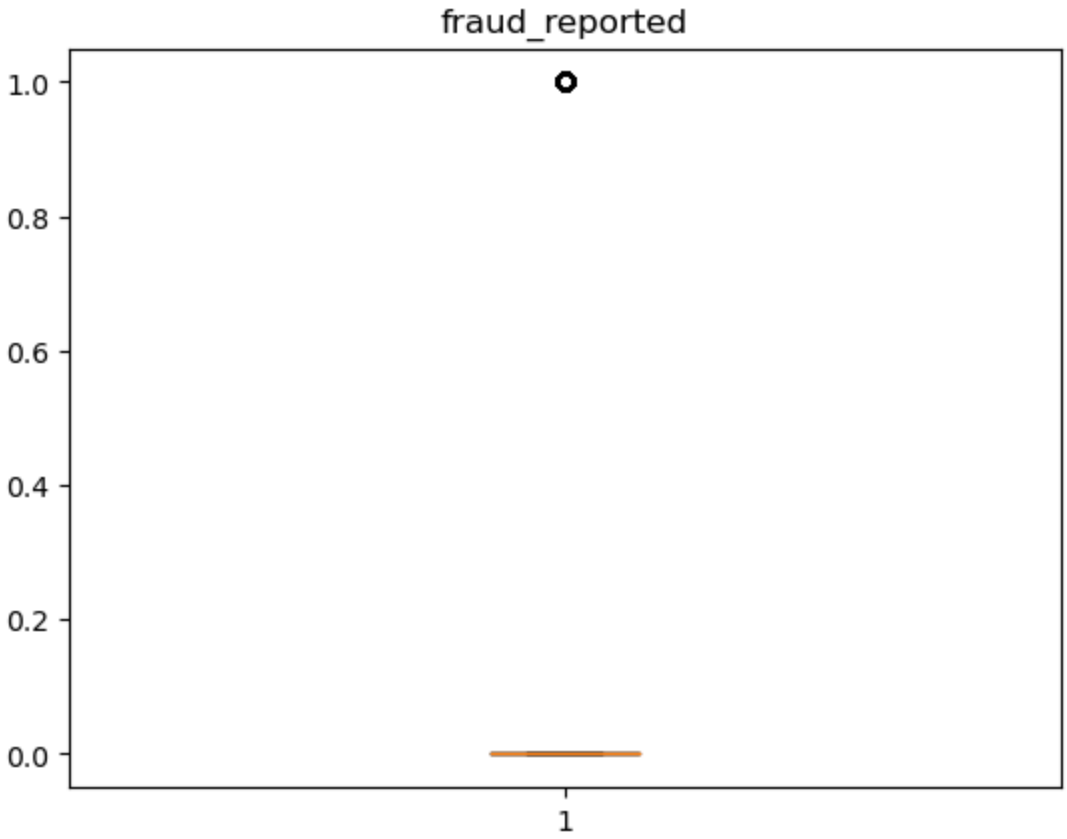

total_claim_amount
120000
100000
80000
60000
40000
20000
0
1

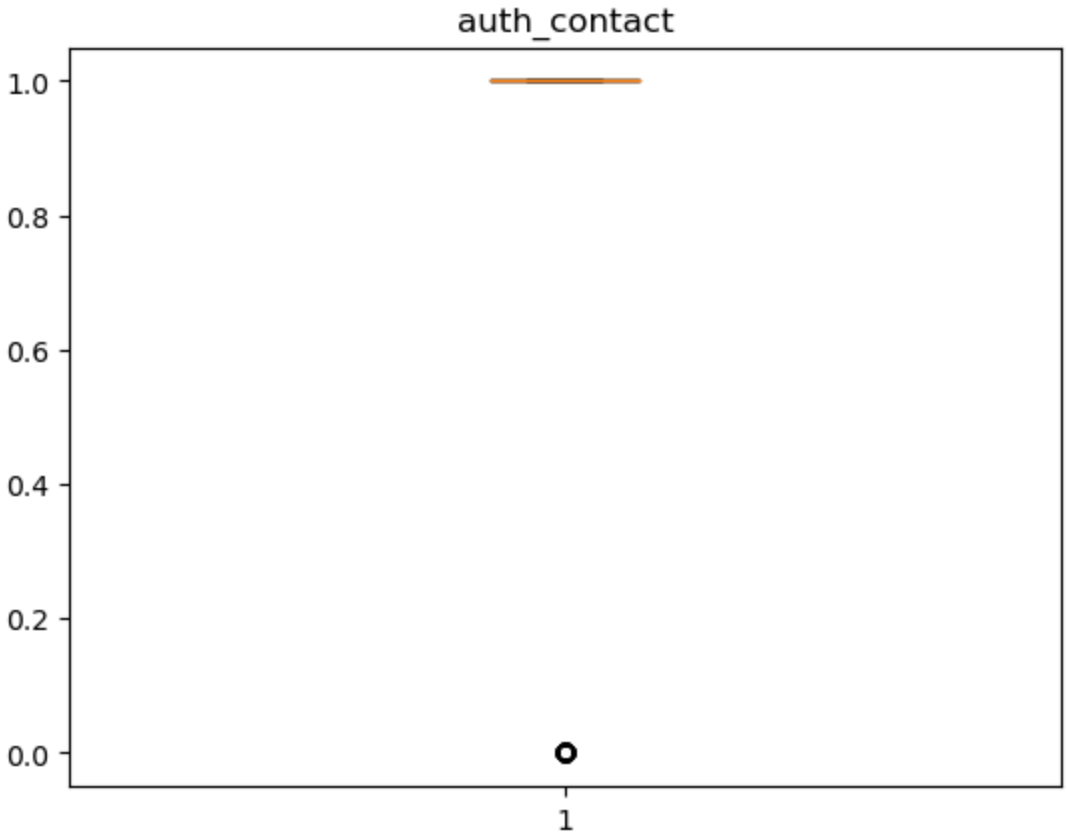

injury_claim
20000
15000
10000
5000
0
1

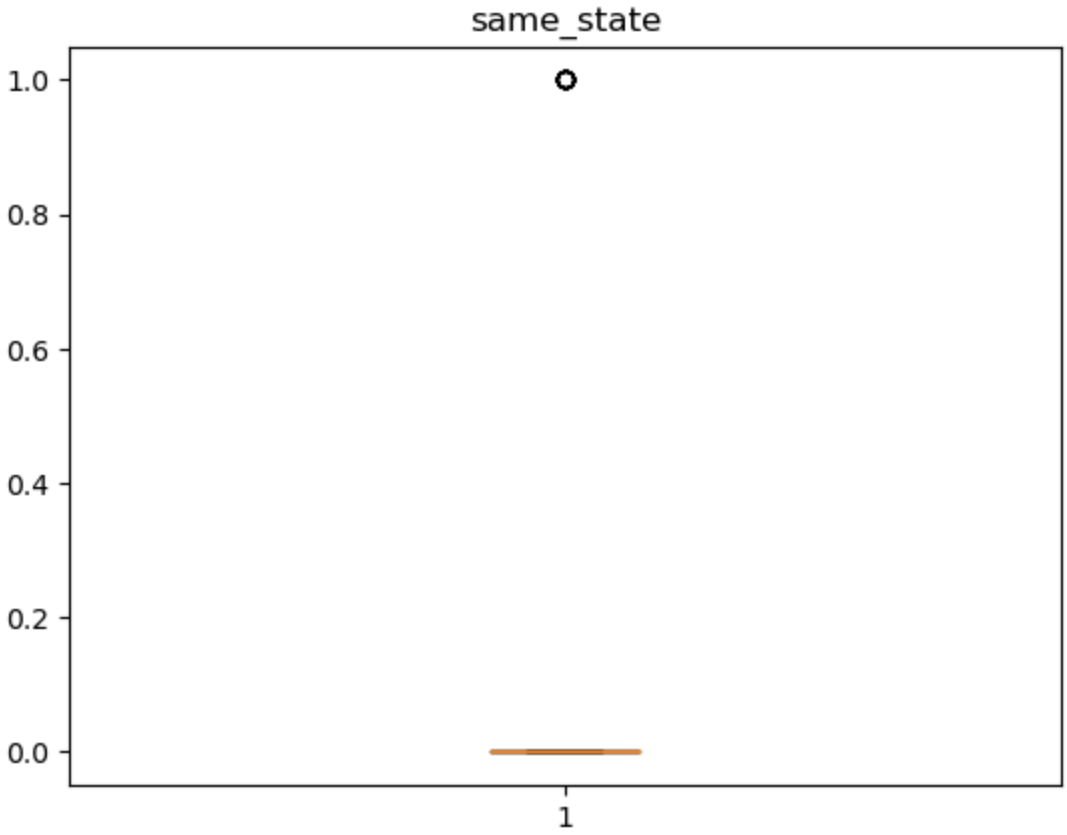

property_claim
20000
15000
10000
5000
0
1


vehicle_claim
80000
70000
60000
50000
40000
30000
20000
10000
0
1

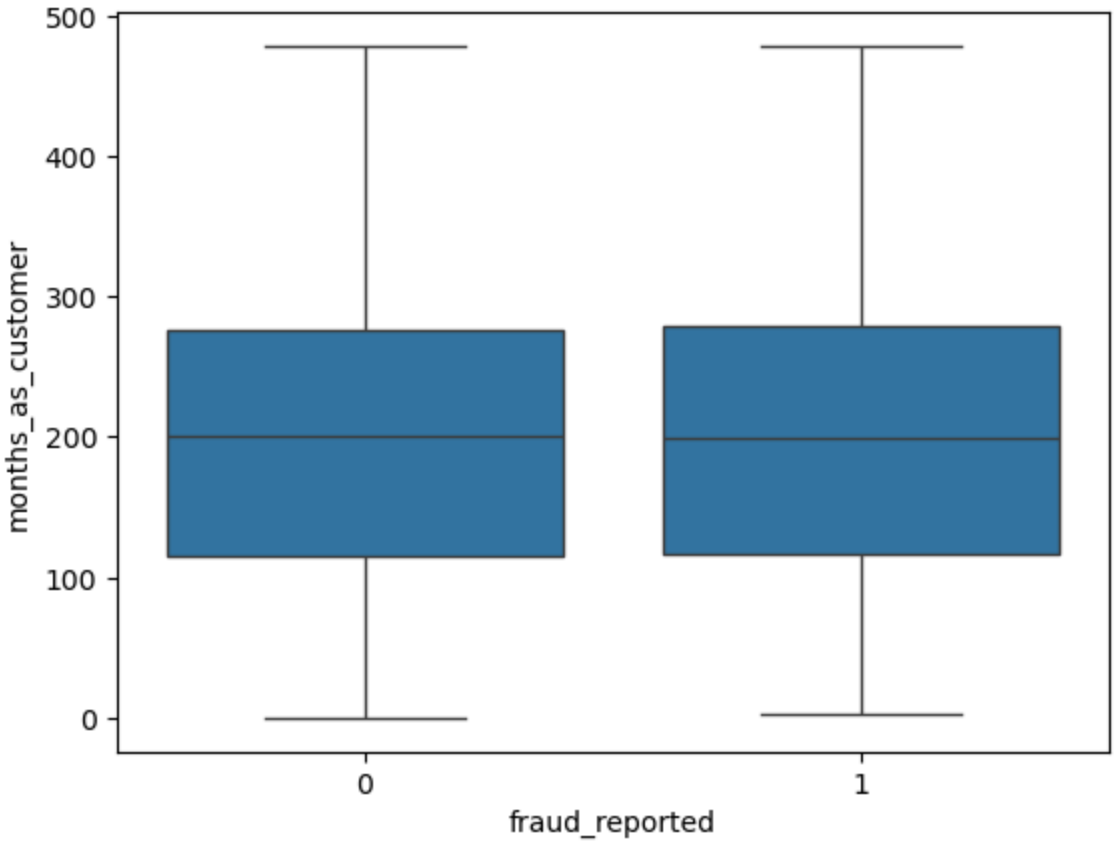

auto_year
2015.0
2012.5
2010.0
2007.5
2005.0
2002.5
2000.0
1997.5
1995.0
1


fraud_reported
1.0
0.8
0.6
0.4
0.2
0.0
1

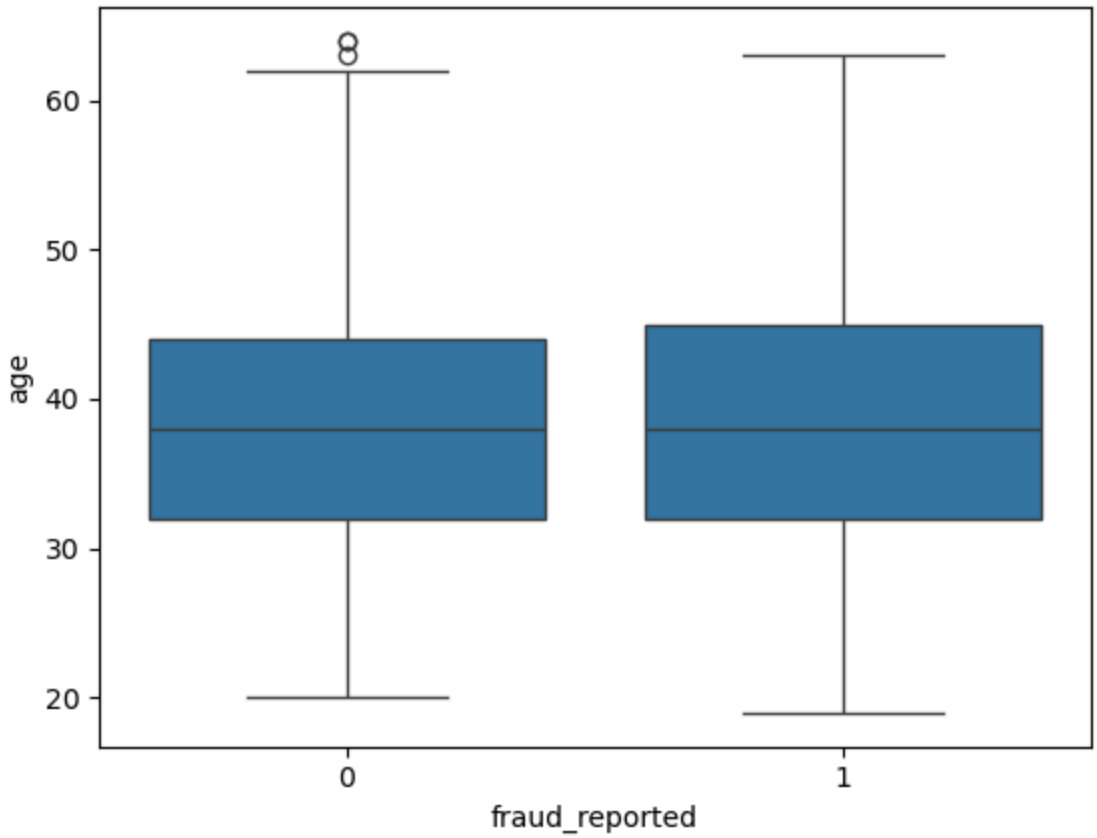

auth_contact
1.0
0.8
0.6
0.4
0.2
0.0
1



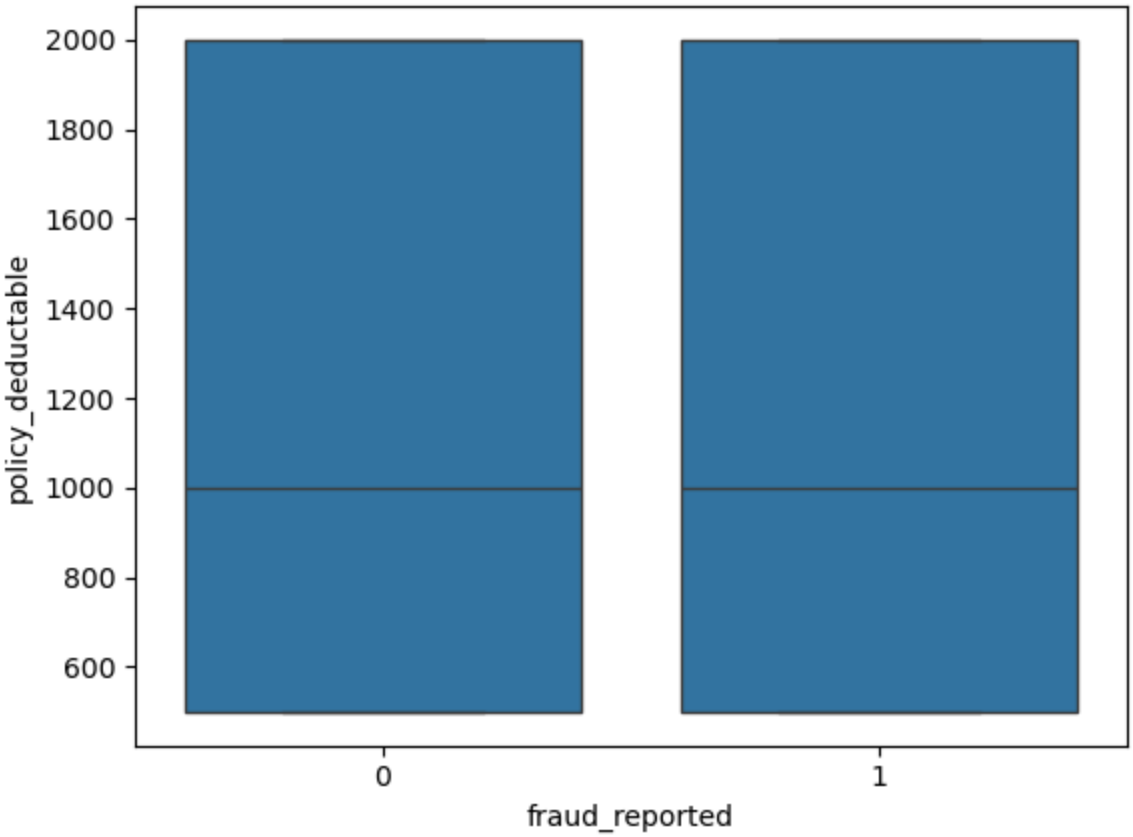
In [30]:
```
ax=sns.boxplot(x="fraud_reported", y="months_as_customer", data=df)
```



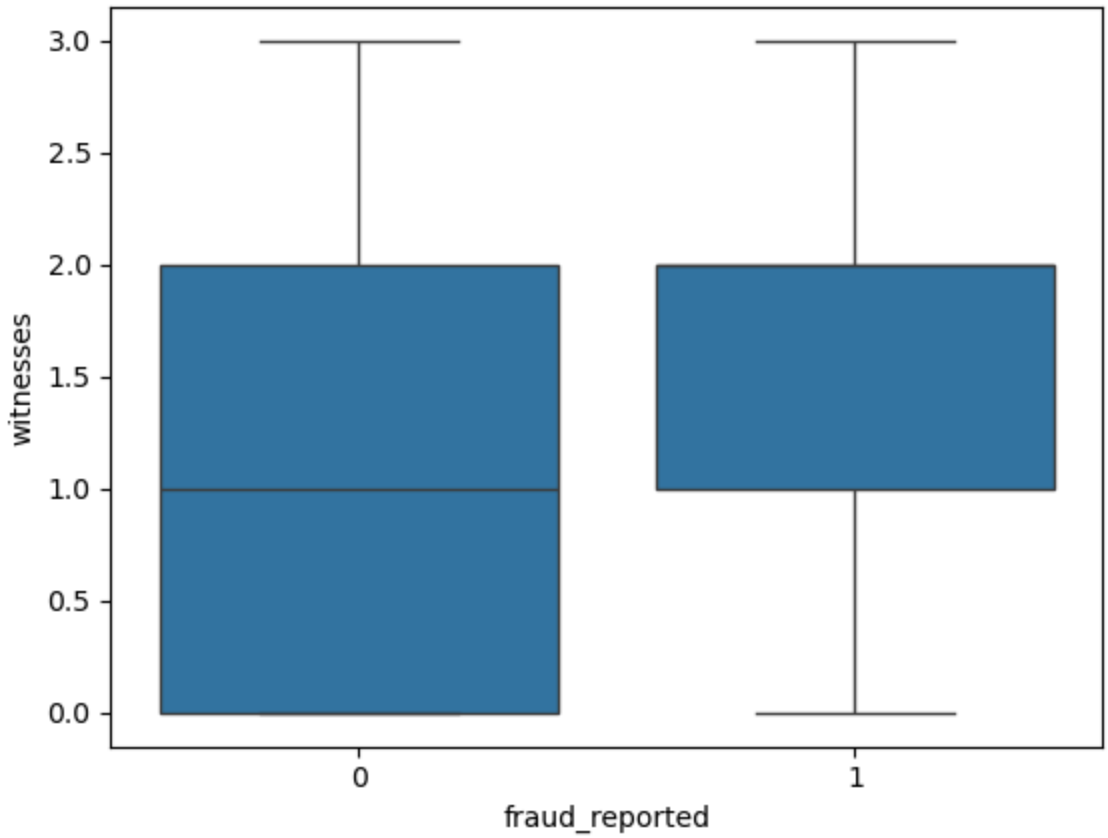
In [31]:
```
ax=sns.boxplot(x="fraud_reported", y="age", data=df)
```



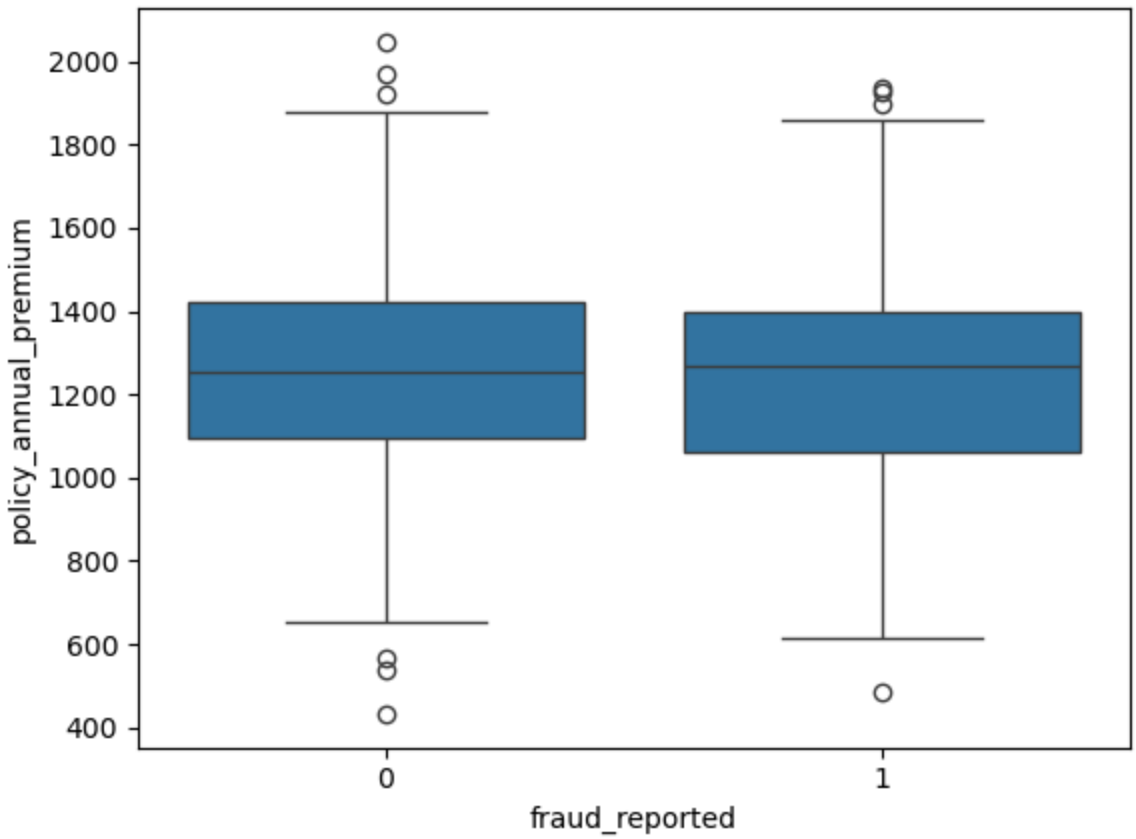
In [32]:
```
ax=sns.boxplot(x="fraud_reported", y="policy_deductable", data=df)
```



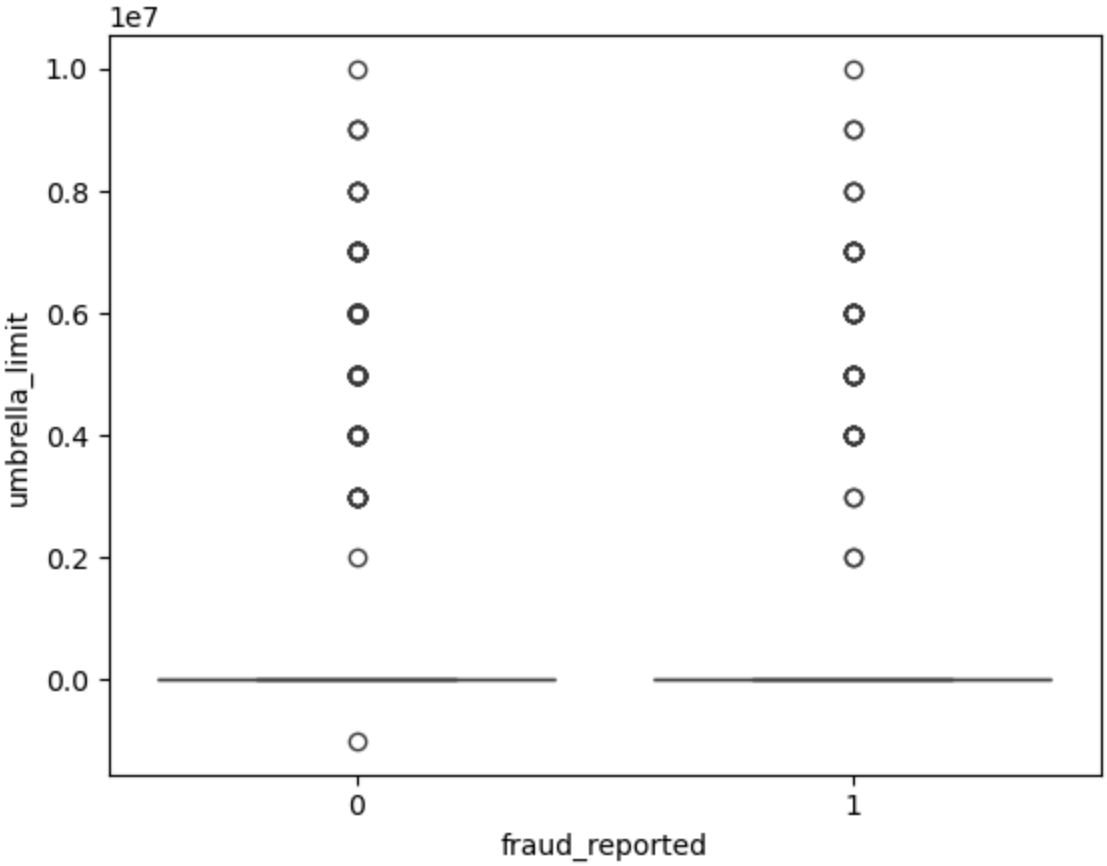
In [33]:
```
ax=sns.boxplot(x="fraud_reported", y="witnesses", data=df)
```



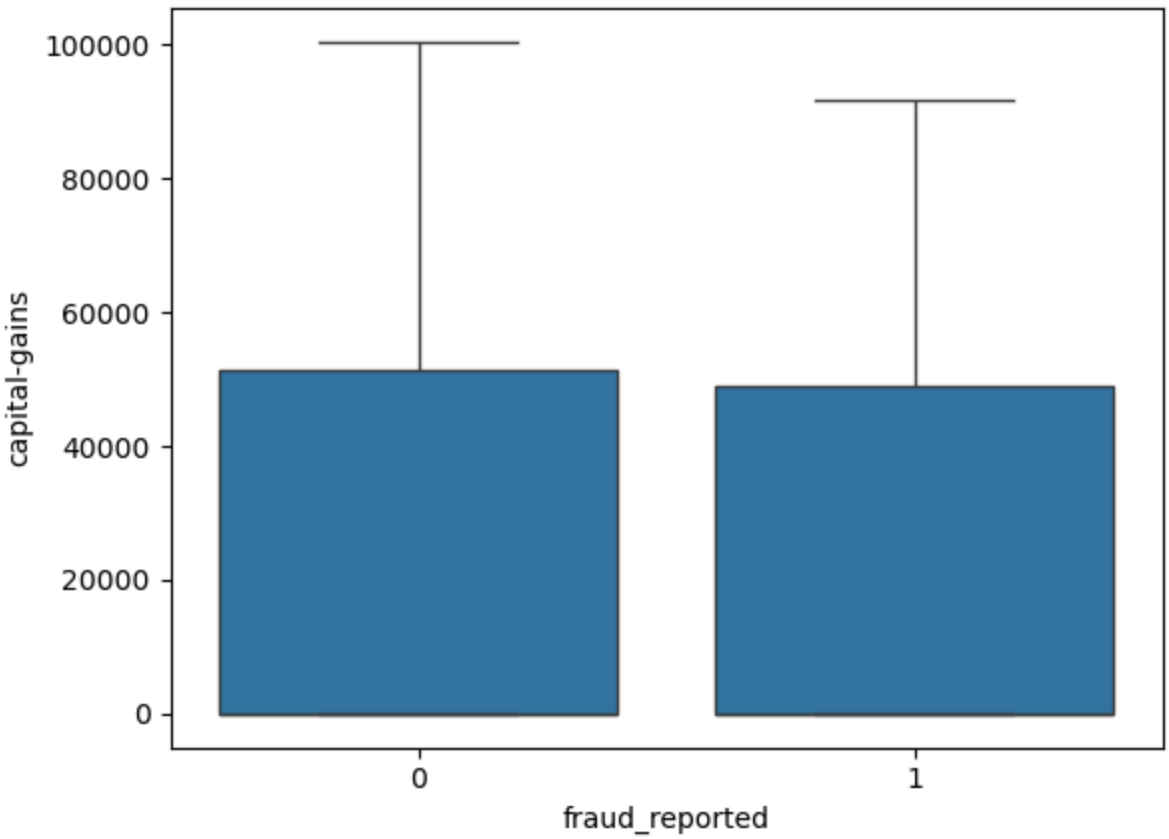
In [34]:
```
ax=sns.boxplot(x="fraud_reported", y="policy_annual_premium", data=df)
```



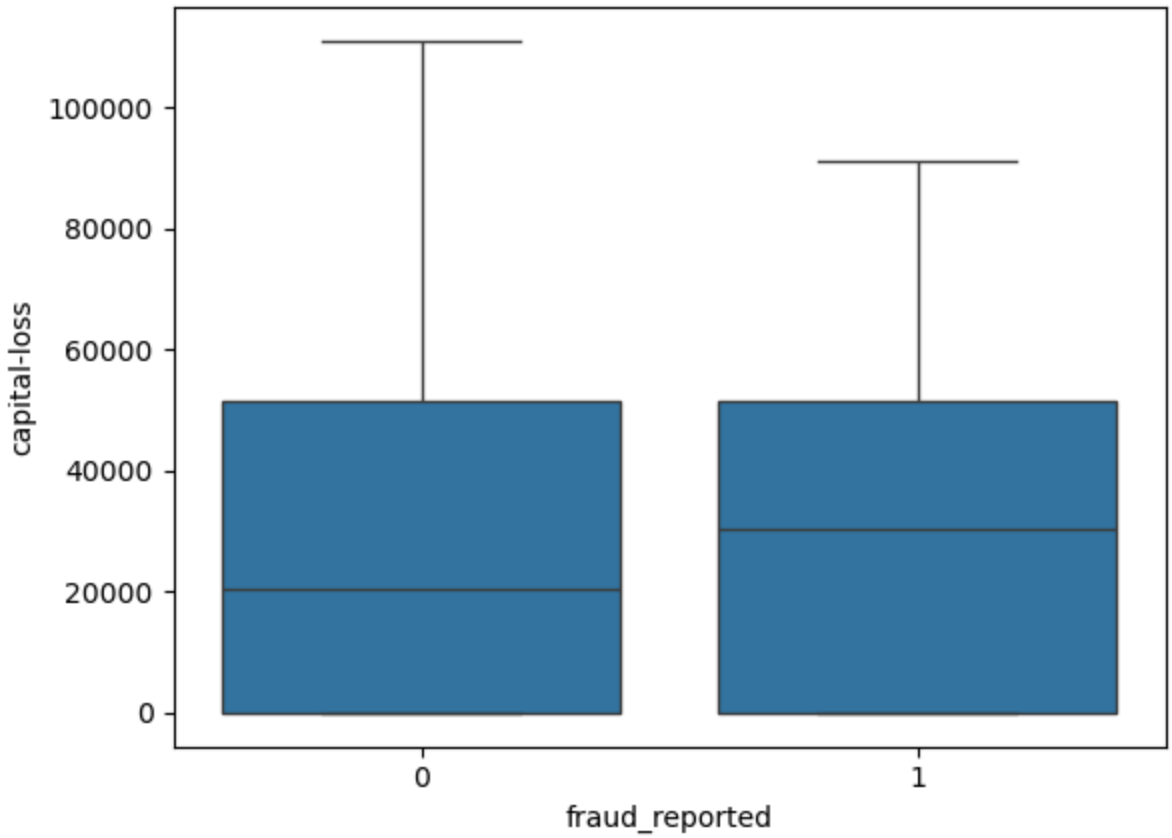
In [35]:
```
ax=sns.boxplot(x="fraud_reported", y="umbrella_limit", data=df)
```



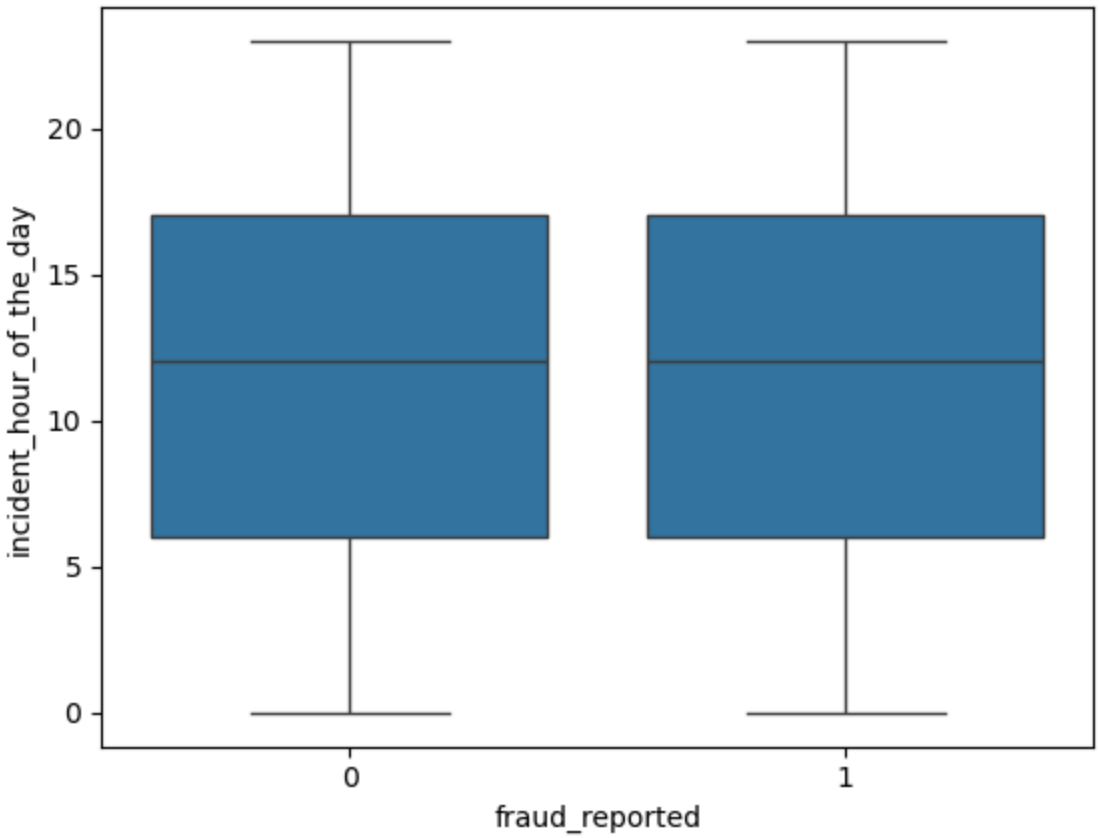
In [36]:
```
ax=sns.boxplot(x="fraud_reported", y="capital-gains", data=df)
```



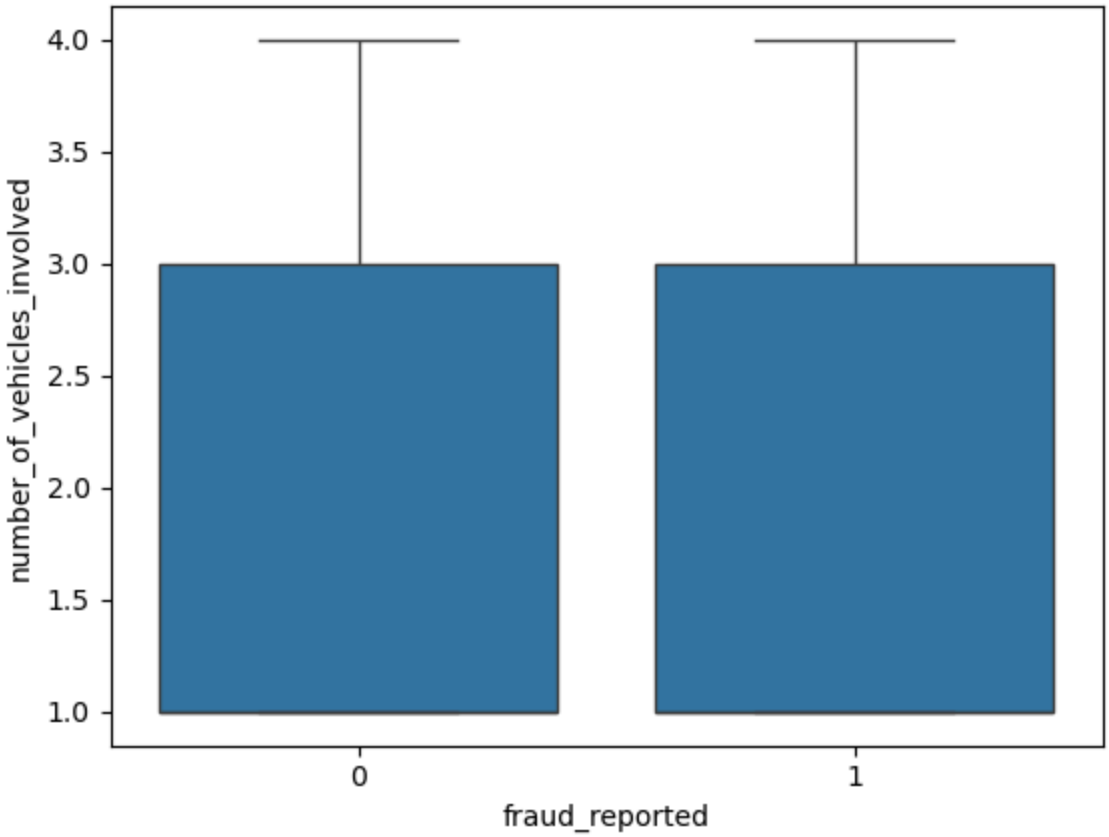
In [37]:
```
ax=sns.boxplot(x="fraud_reported", y="capital-loss", data=df)
```



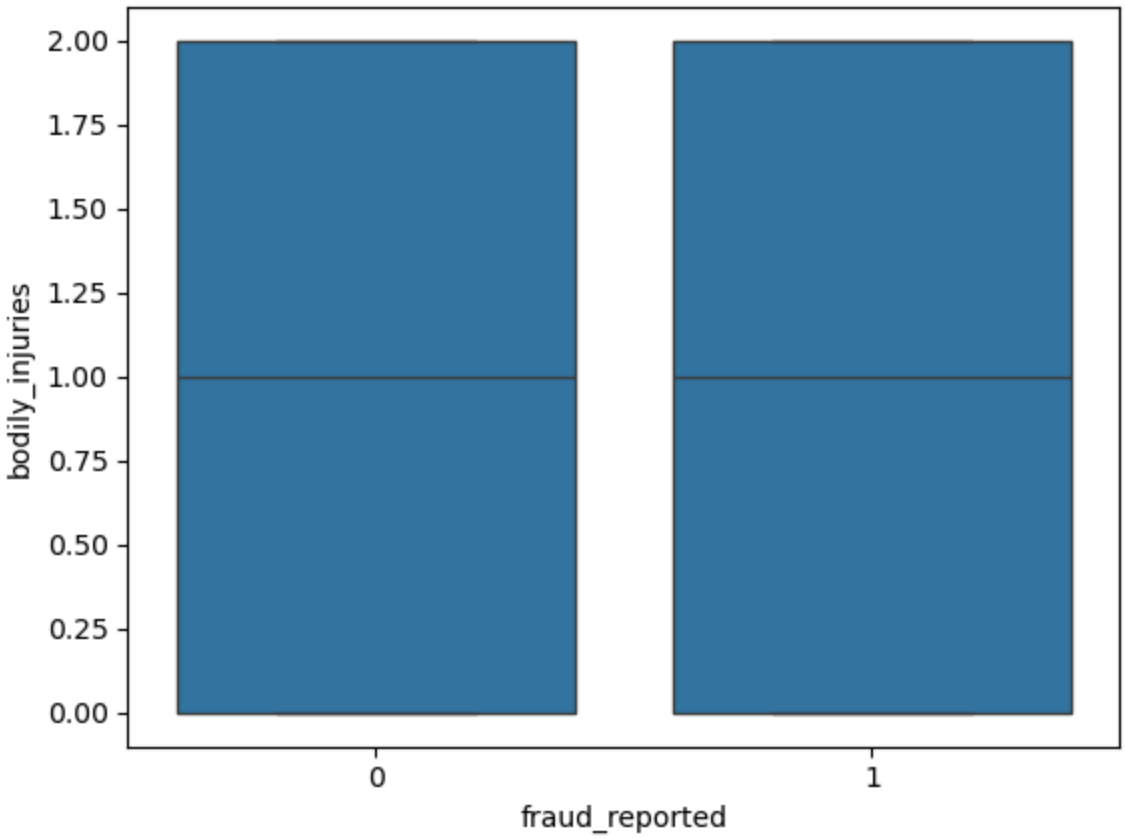
In [38]:
```
ax=sns.boxplot(x="fraud_reported", y="incident_hour_of_the_day", data=df)
```



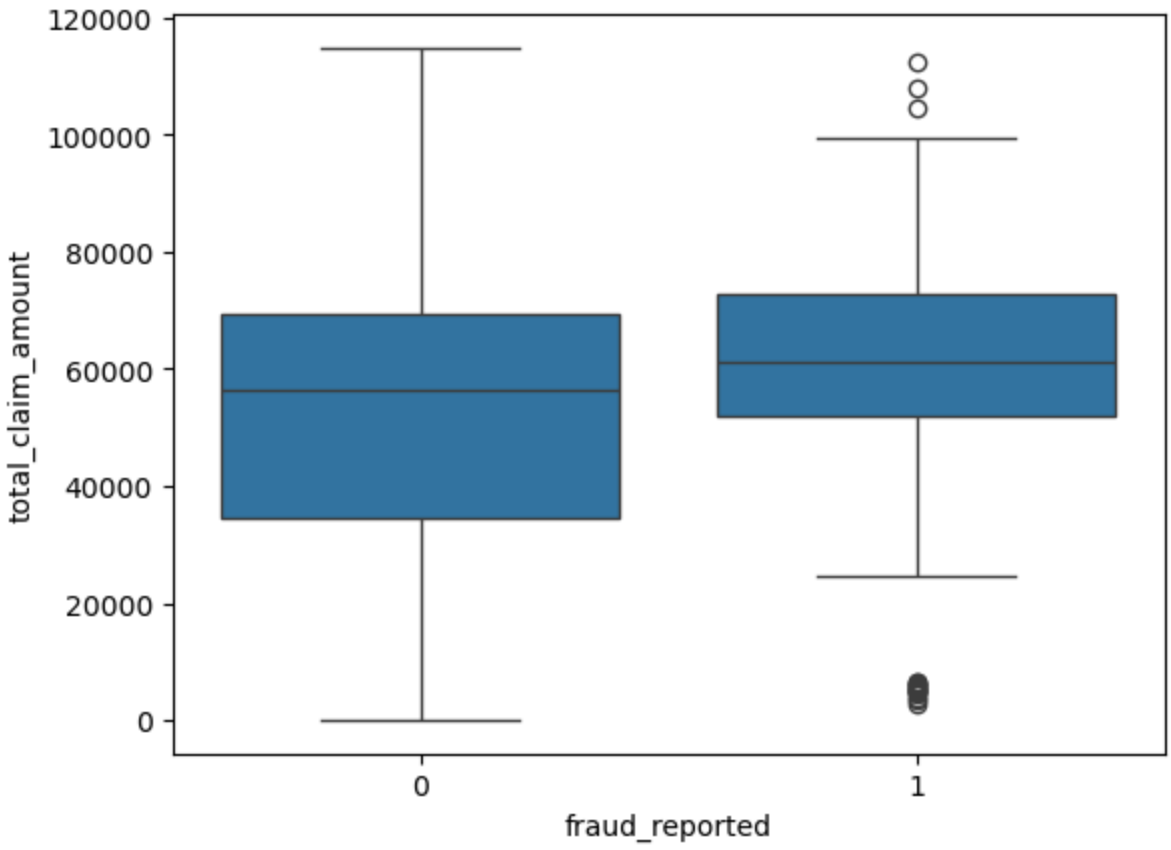
In [39]:
```
ax=sns.boxplot(x="fraud_reported", y="number_of_vehicles_involved", data=df)
```



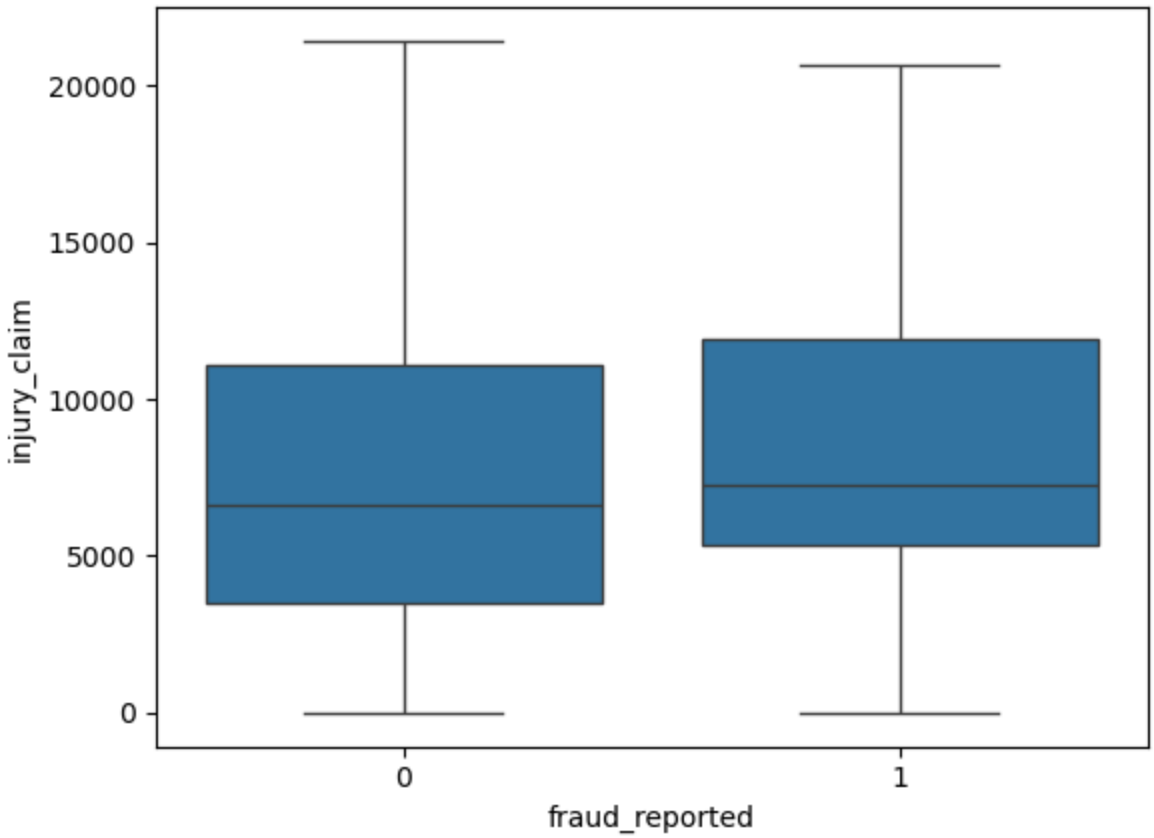
In [40]:
```
ax=sns.boxplot(x="fraud_reported", y="bodily_injuries", data=df)
```



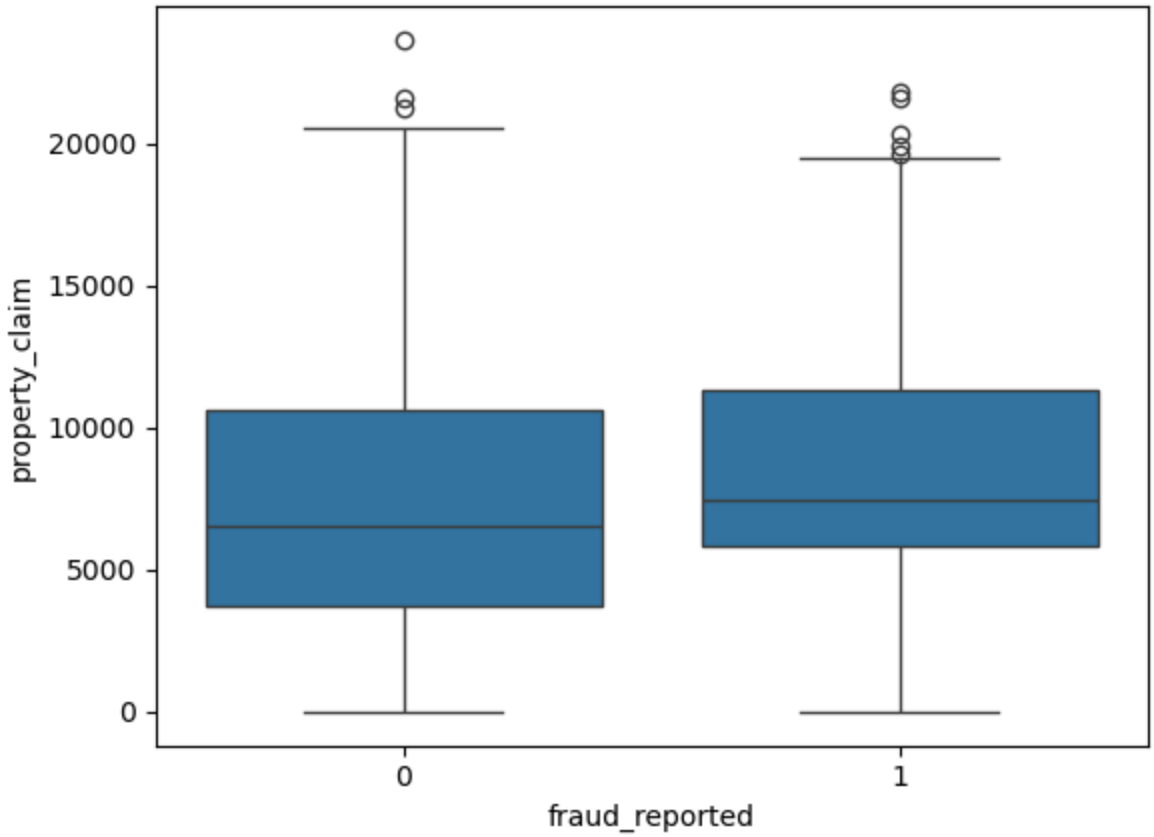
In [41]:
```
ax=sns.boxplot(x="fraud_reported", y="total_claim_amount", data=df)
```



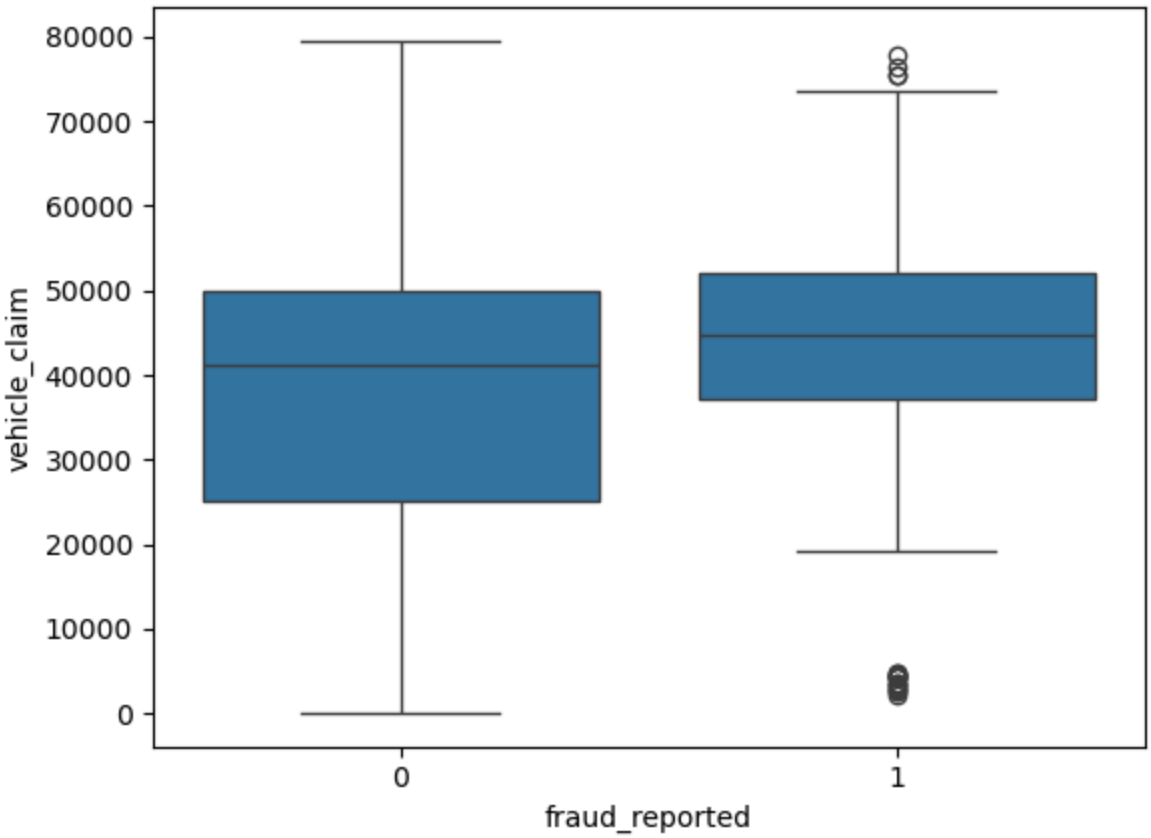
In [42]:
```
ax=sns.boxplot(x="fraud_reported", y="injury_claim", data=df)
```



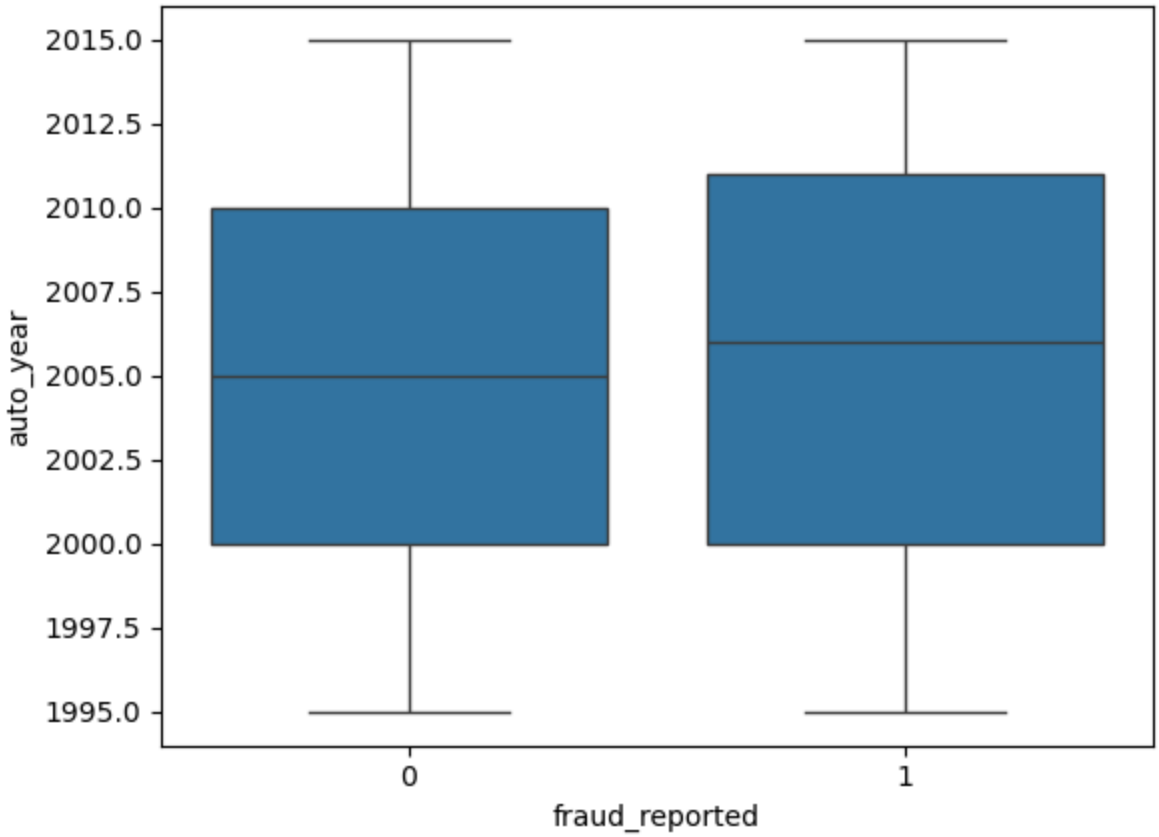
In [43]:
```
ax=sns.boxplot(x="fraud_reported", y="property_claim", data=df)
```



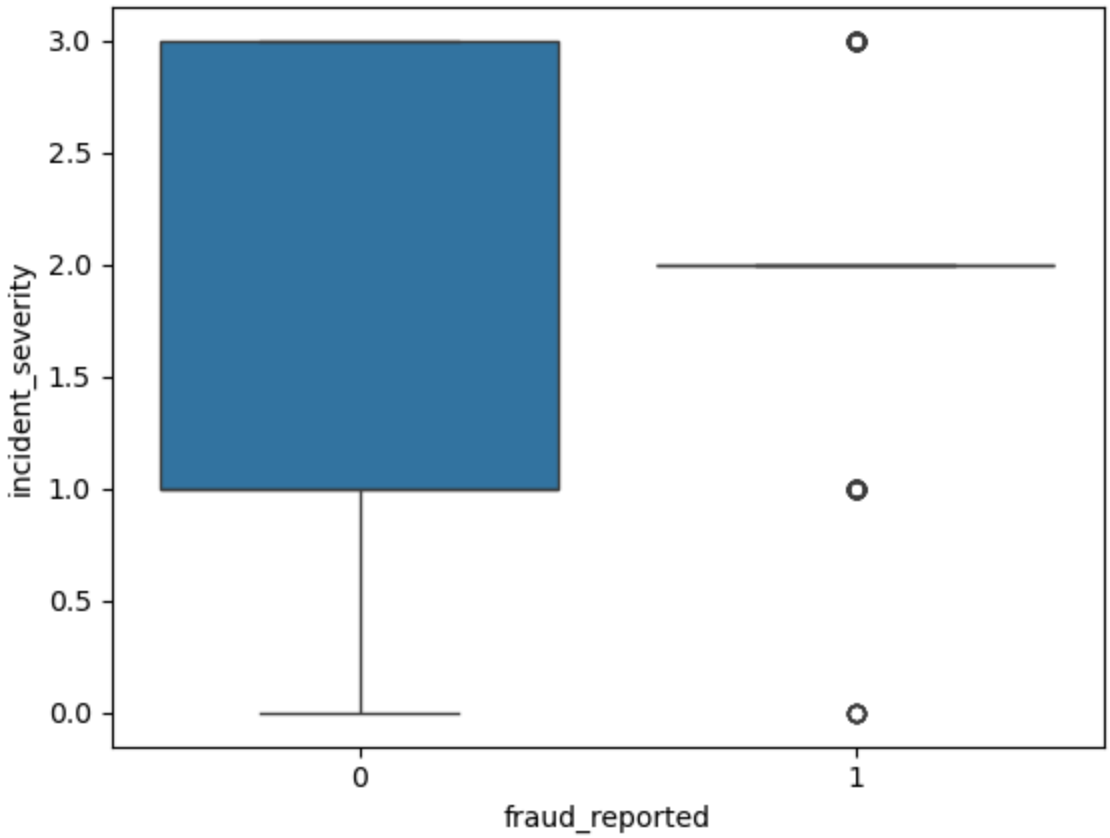
In [44]:
```
ax=sns.boxplot(x="fraud_reported", y="vehicle_claim", data=df)
```



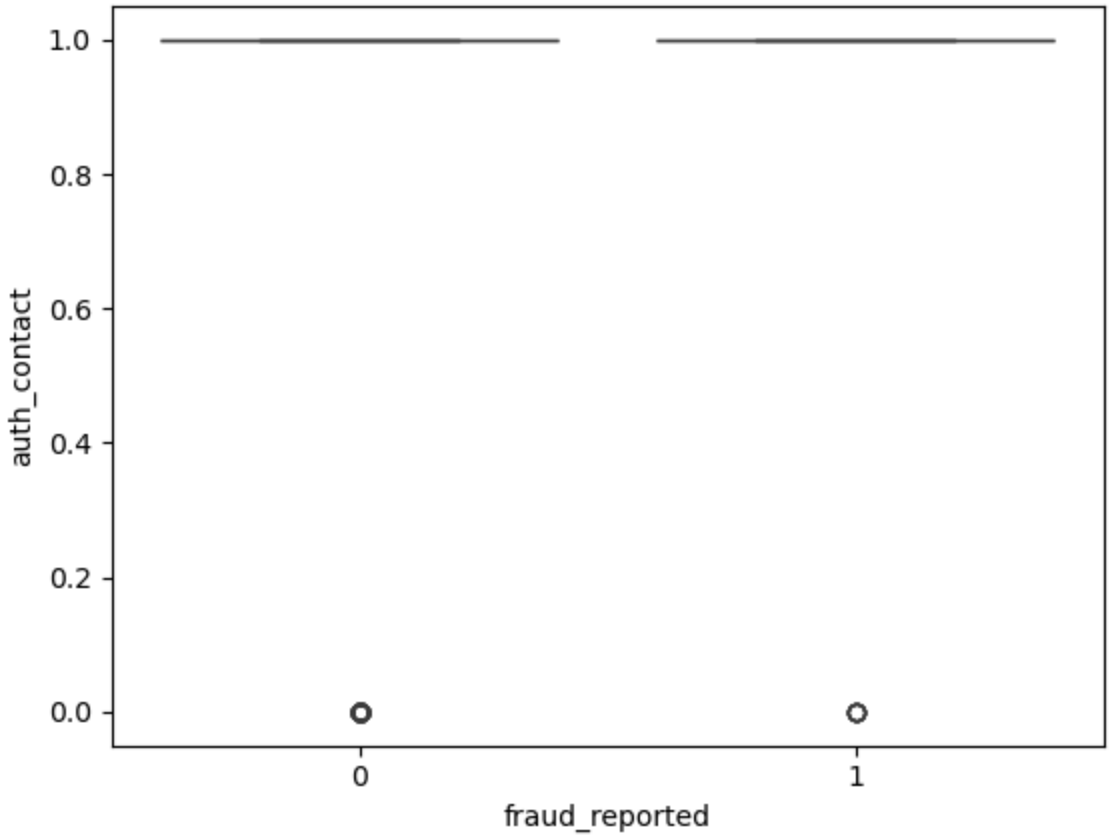
In [45]:
```
ax=sns.boxplot(x="fraud_reported", y="auto_year", data=df)
```



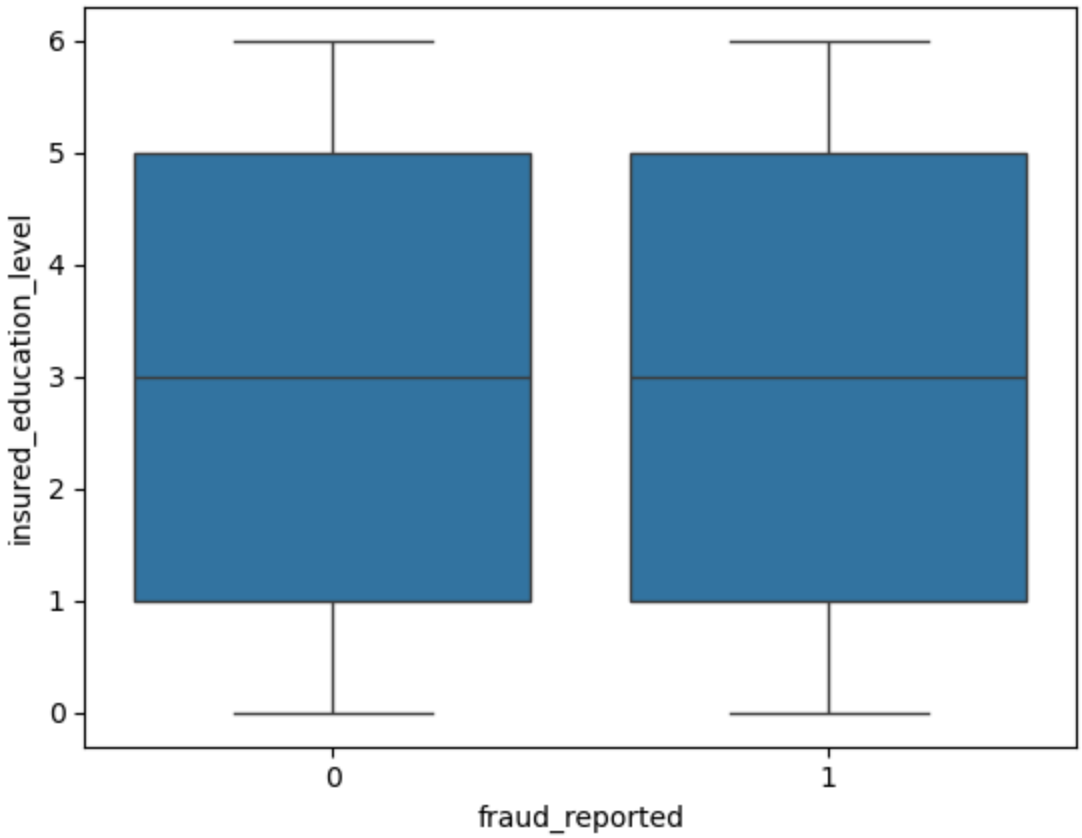
In [46]:
```
ax=sns.boxplot(x="fraud_reported", y="incident_severity", data=df)
```



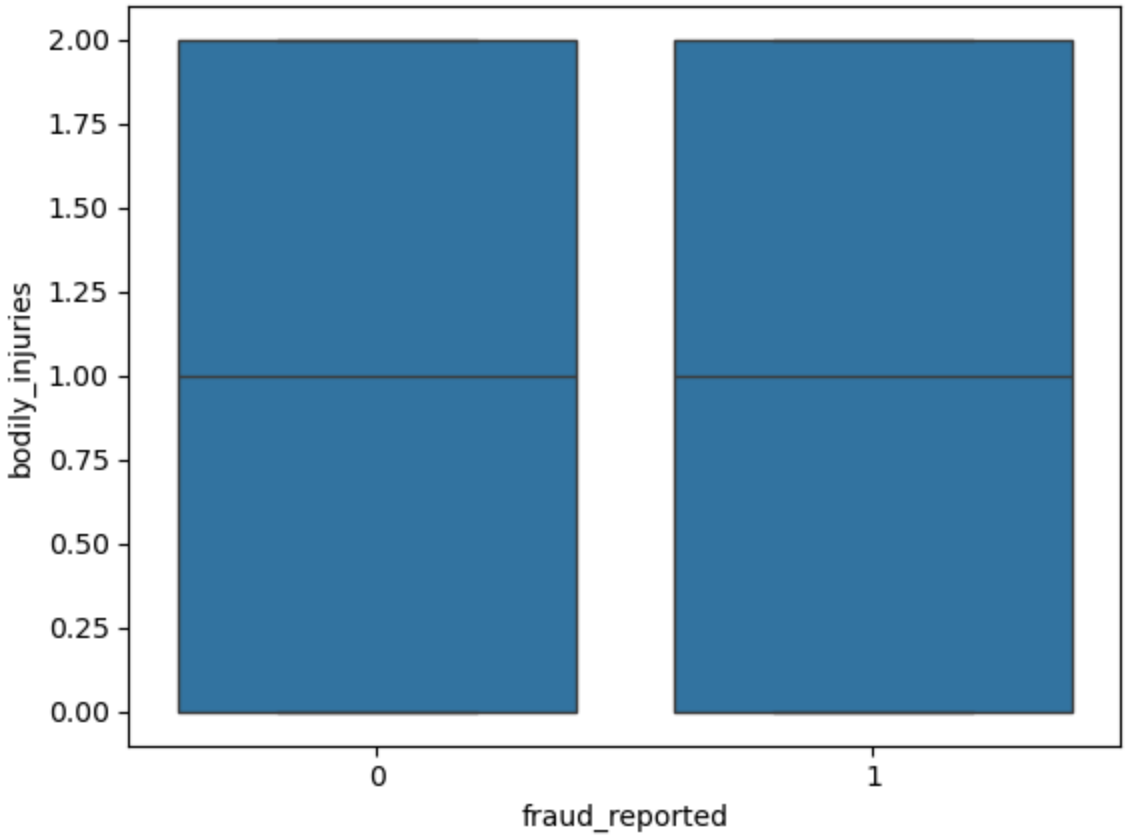
In [47]:
```
ax=sns.boxplot(x="fraud_reported", y="auth_contact", data=df)
```



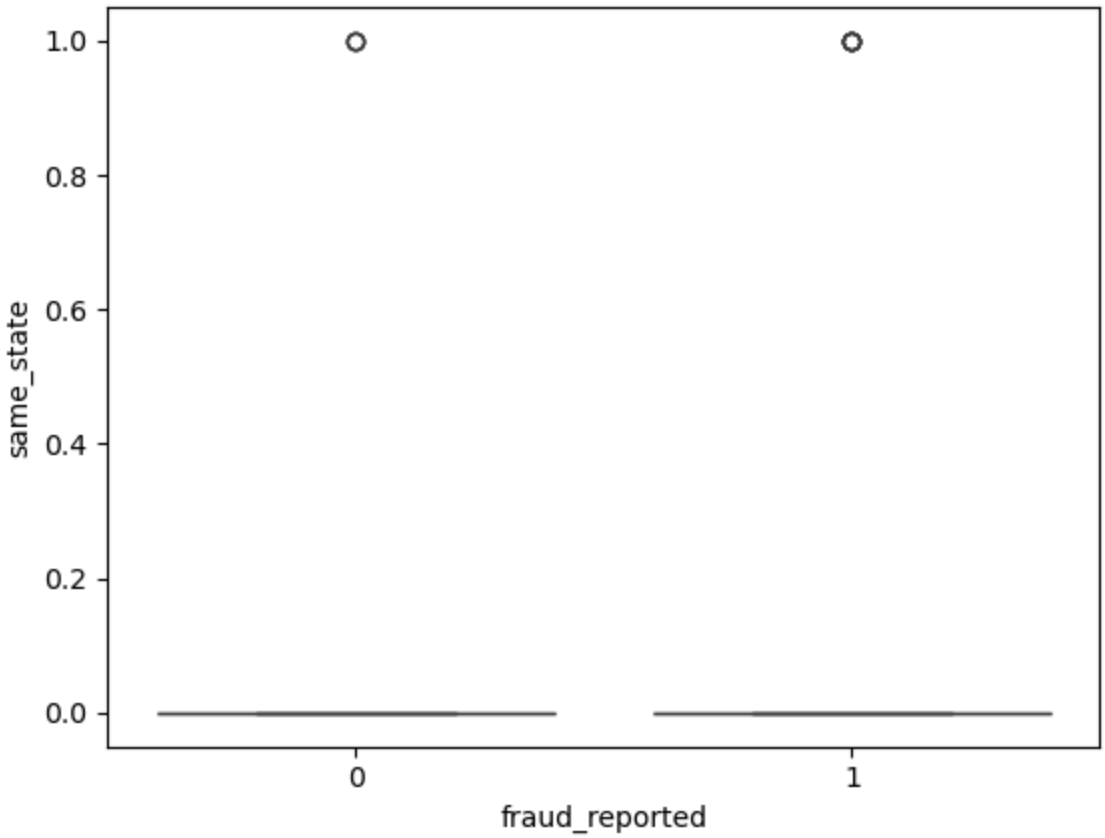
In [48]:
```
ax=sns.boxplot(x="fraud_reported", y="insured_education_level", data=df)
```



In [49]:
```
ax=sns.boxplot(x="fraud_reported", y="bodily_injuries", data=df)
```



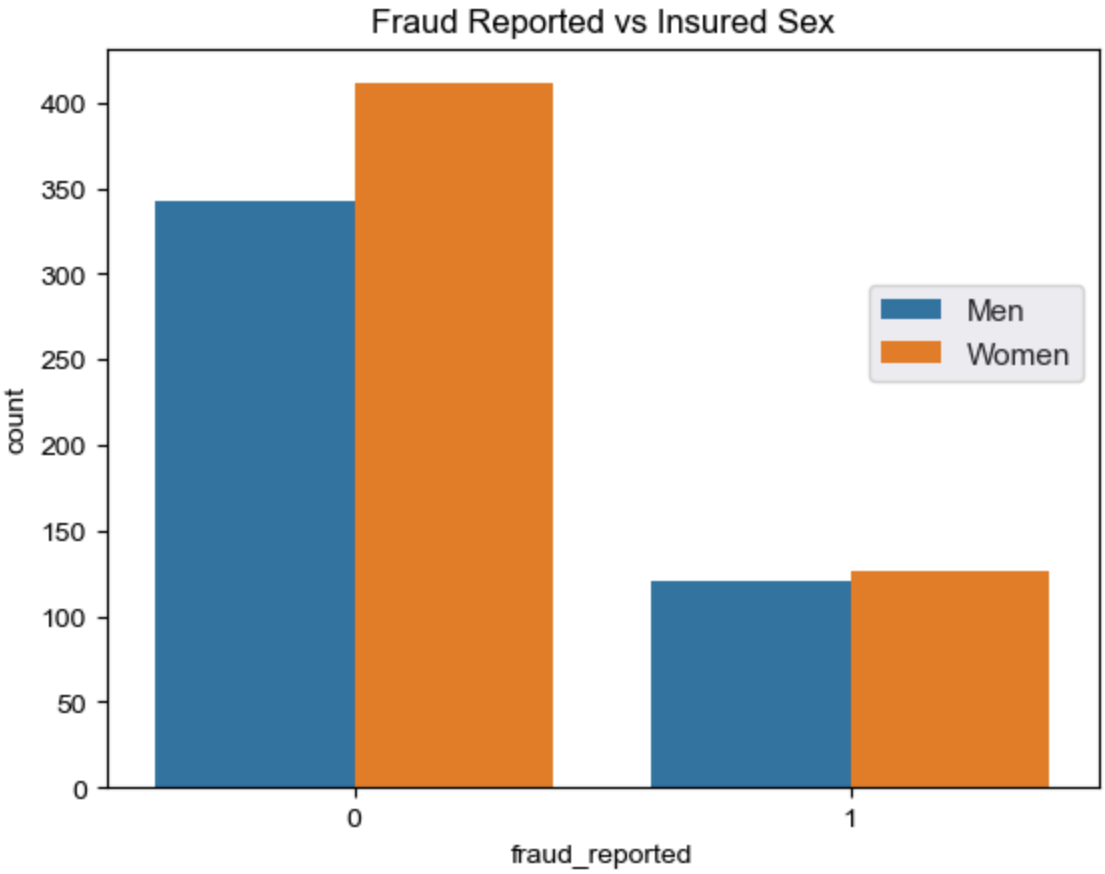
In [50]:
```
ax=sns.boxplot(x="fraud_reported", y="same_state", data=df)
```



In [51]:
```
#Looking for interesting relationships
```

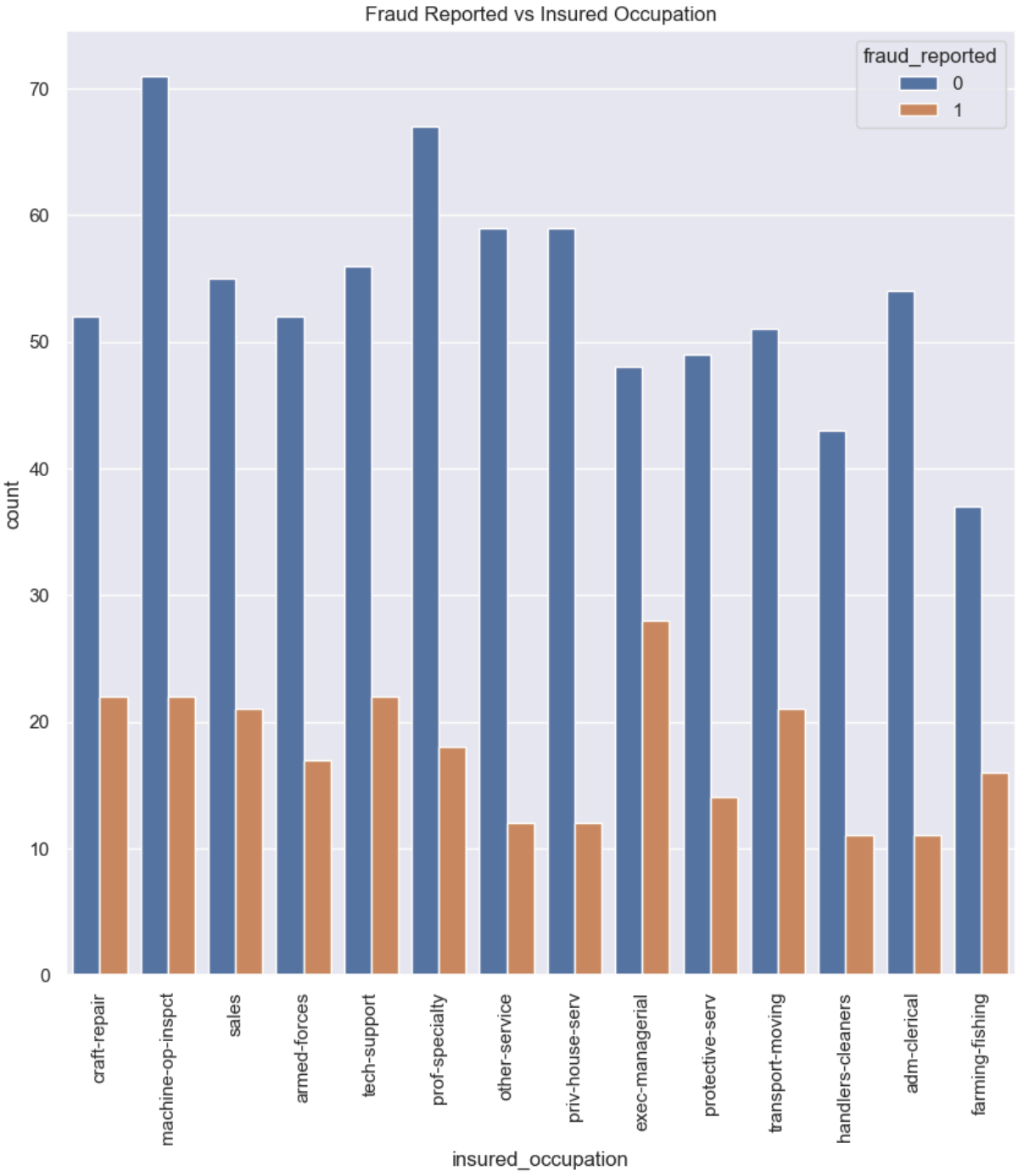
In [52]:
```
sns.countplot(x='fraud_reported',hue='insured_sex',data=df)
sns.set(rc={'figure.figsize':(10,10)})
plt.legend(['Men', 'Women'], bbox_to_anchor=(1.0, 0.7))
plt.title("Fraud Reported vs Insured Sex")
```

Out[52]:
```
Text(0.5, 1.0, 'Fraud Reported vs Insured Sex')
```



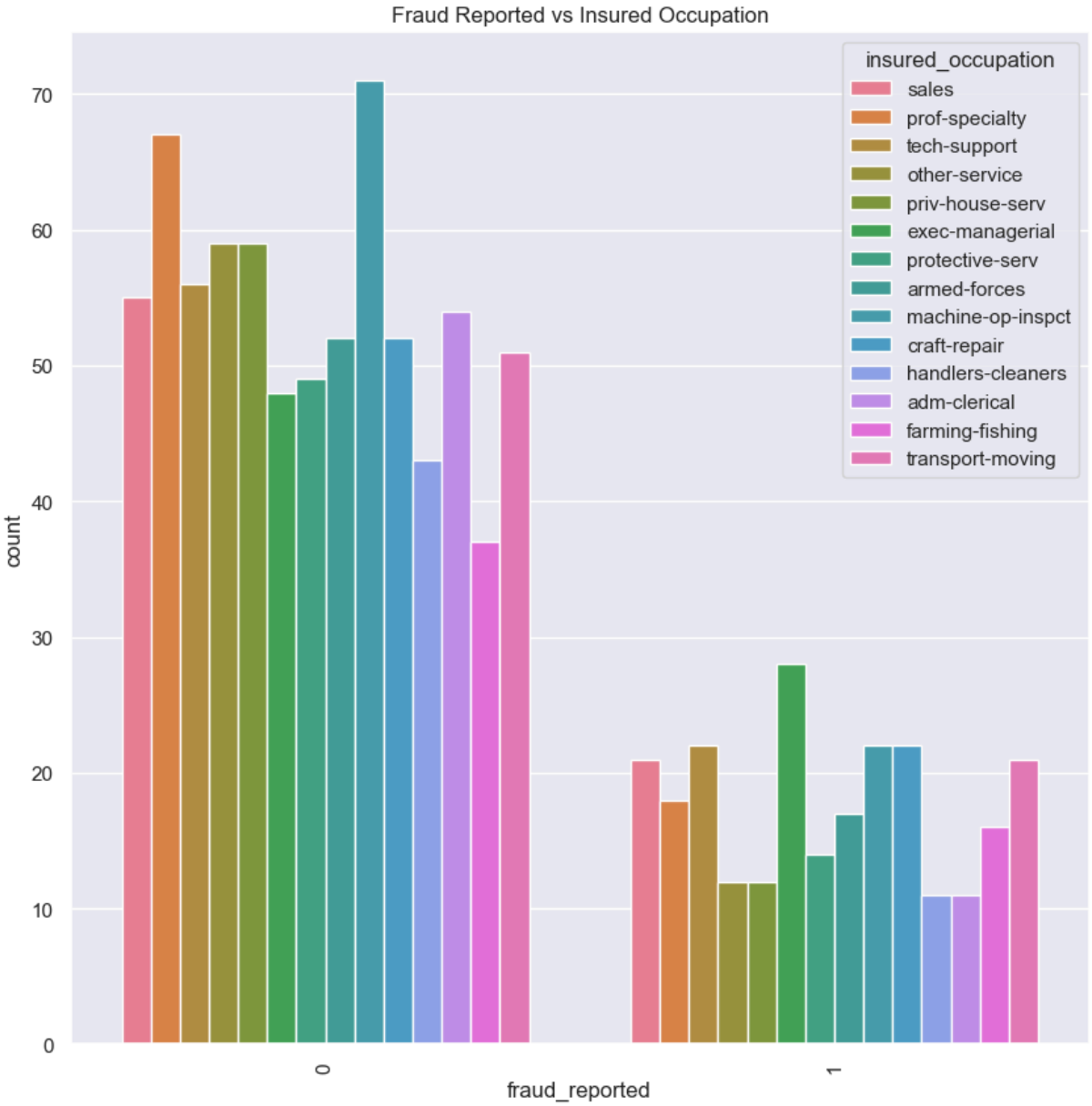
In [53]:
```
sns.countplot(x='insured_occupation',hue='fraud_reported',data=df)
sns.set(rc={'figure.figsize':(10,10)})
plt.xticks(rotation=90)
plt.title("Fraud Reported vs Insured Occupation")
```

Out[53]:
```
Text(0.5, 1.0, 'Fraud Reported vs Insured Occupation')
```



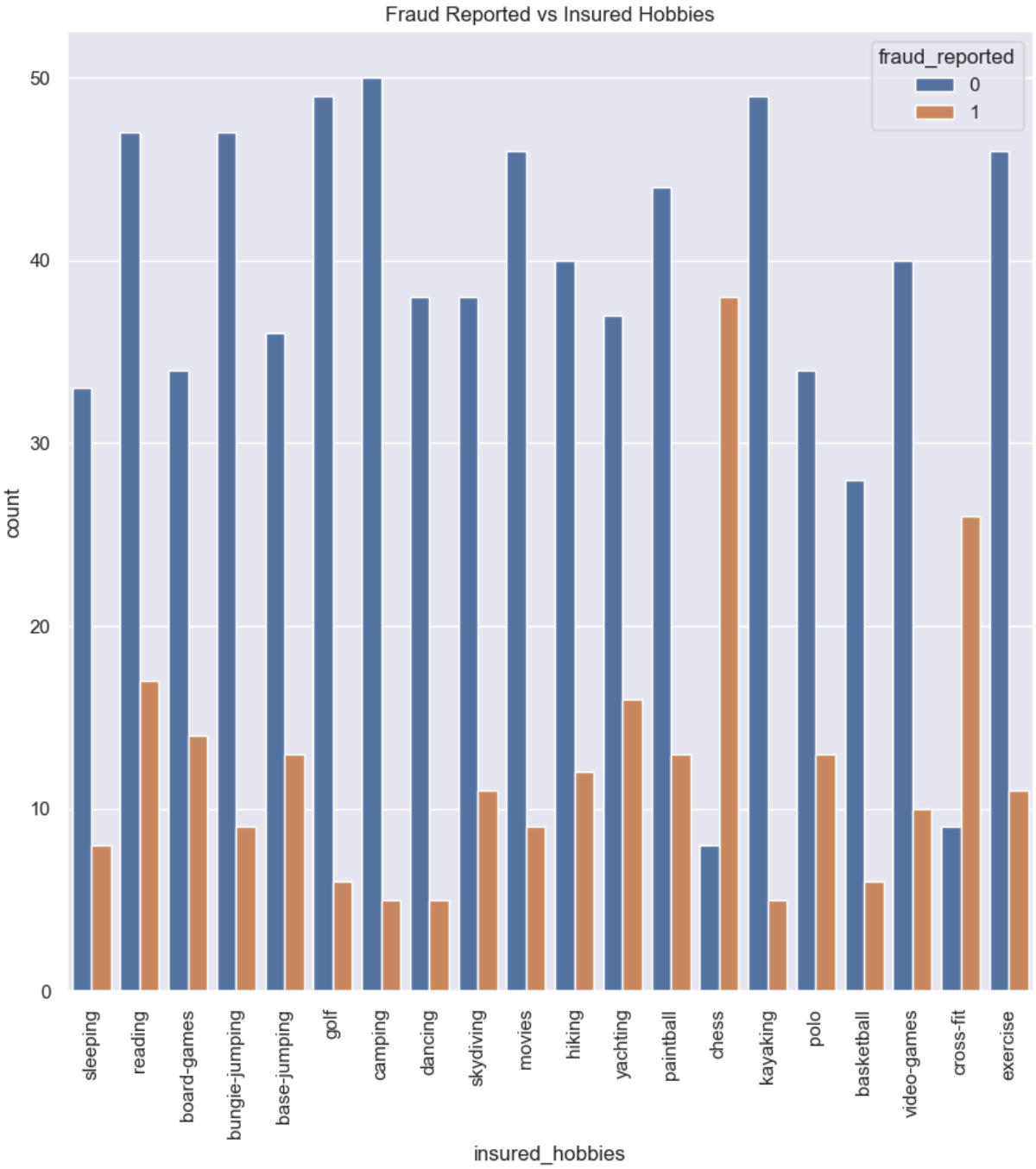
In [54]:
```
sns.countplot(x='fraud_reported',hue='insured_occupation',data=df)
sns.set(rc={'figure.figsize':(10,10)})
plt.xticks(rotation=90)
plt.title("Fraud Reported vs Insured Occupation")
```

Out[54]:
```
Text(0.5, 1.0, 'Fraud Reported vs Insured Occupation')
```



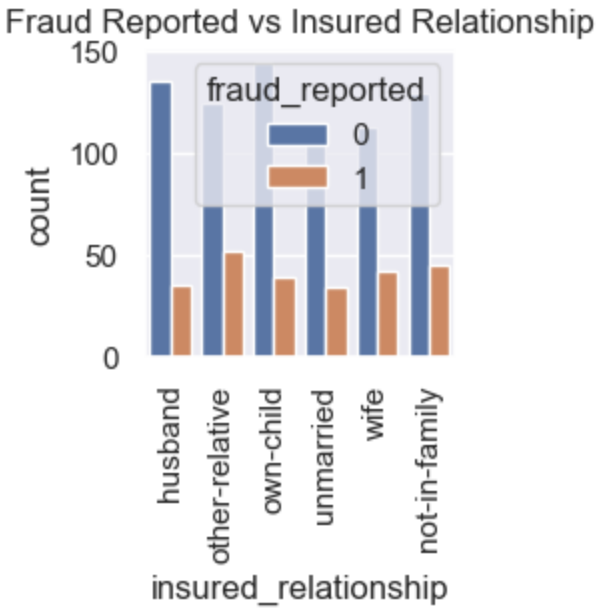
In [55]:
```
sns.countplot(x='insured_hobbies',hue='fraud_reported',data=df)
sns.set(rc={'figure.figsize':(2,2)})
plt.xticks(rotation=90)
plt.title("Fraud Reported vs Insured Hobbies")
```

Out[55]:
```
Text(0.5, 1.0, 'Fraud Reported vs Insured Hobbies')
```



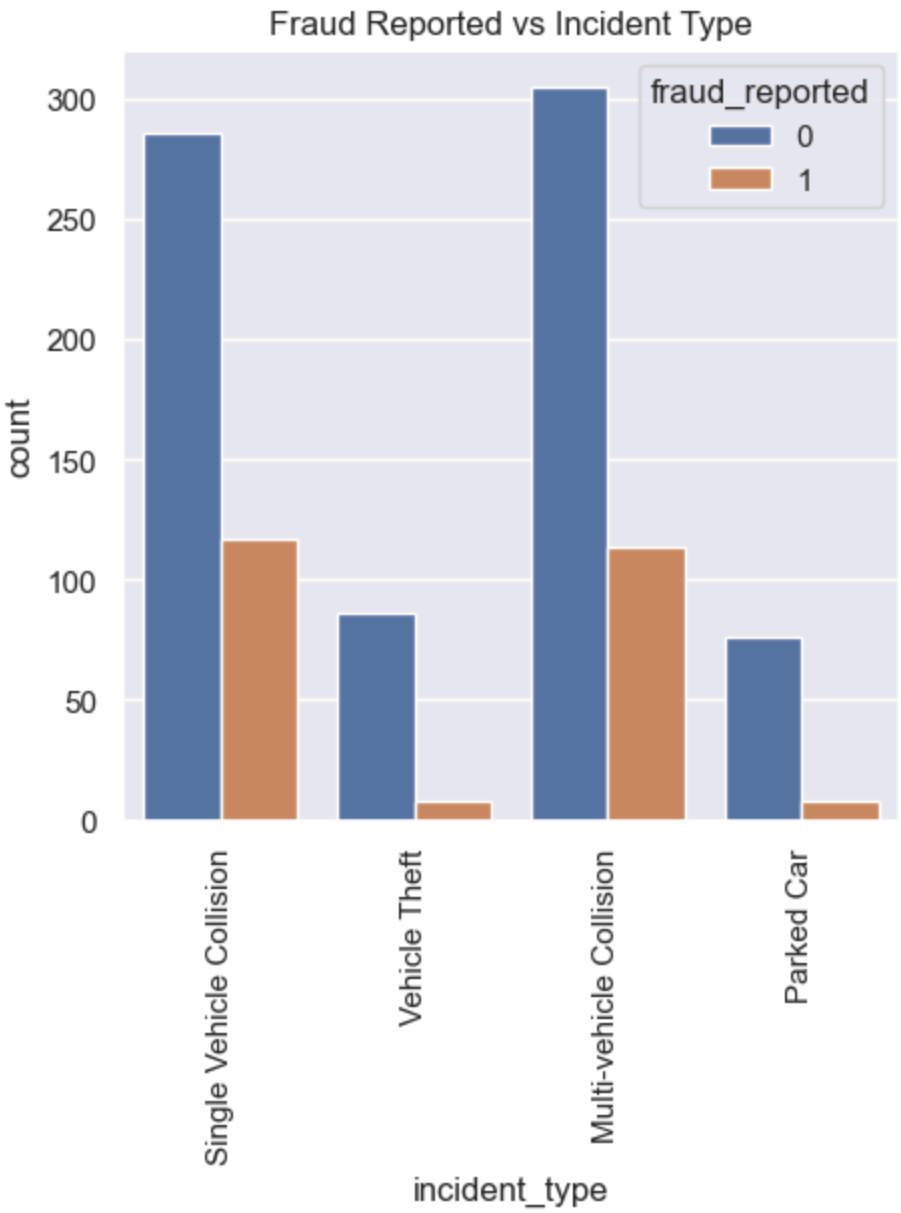
In [56]:
```
sns.countplot(x='insured_relationship',hue='fraud_reported',data=df)
sns.set(rc={'figure.figsize':(5,5)})
plt.xticks(rotation=90)
plt.title("Fraud Reported vs Insured Relationship")
```

Out[56]:
```
Text(0.5, 1.0, 'Fraud Reported vs Insured Relationship')
```



In [57]:
```
sns.countplot(x='incident_type',hue='fraud_reported',data=df)
sns.set(rc={'figure.figsize':(5,5)})
plt.xticks(rotation=90)
plt.title("Fraud Reported vs Incident Type")
```

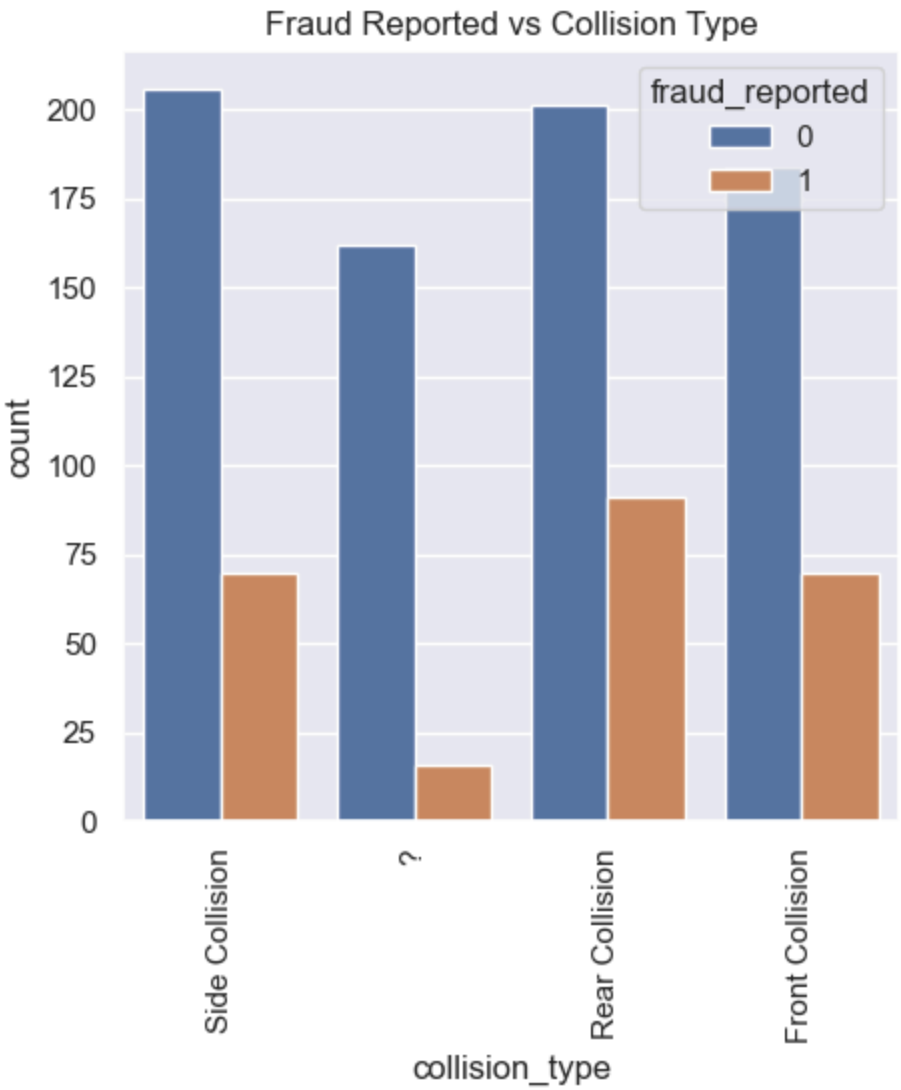
Out[57]:
```
Text(0.5, 1.0, 'Fraud Reported vs Incident Type')
```



In [ ]:

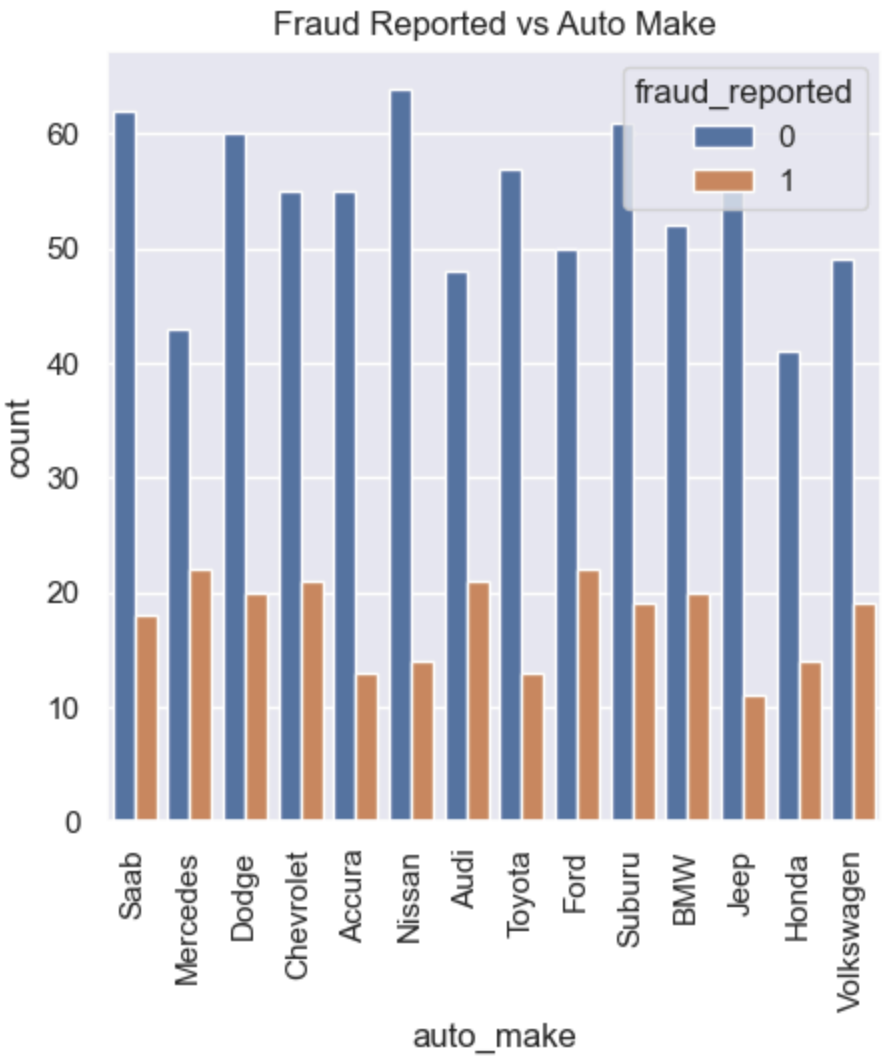
In [58]:
```
sns.countplot(x='collision_type',hue='fraud_reported',data=df)
sns.set(rc={'figure.figsize':(5,5)})
plt.xticks(rotation=90)
plt.title("Fraud Reported vs Collision Type")
```

Out[58]: Text(0.5, 1.0, 'Fraud Reported vs Collision Type')



In [59]:
```
sns.countplot(x='auto_make',hue='fraud_reported',data=df)
sns.set(rc={'figure.figsize':(5,5)})
plt.xticks(rotation=90)
plt.title("Fraud Reported vs Auto Make")
```

Out[59]:
```
Text(0.5, 1.0, 'Fraud Reported vs Auto Make')
```

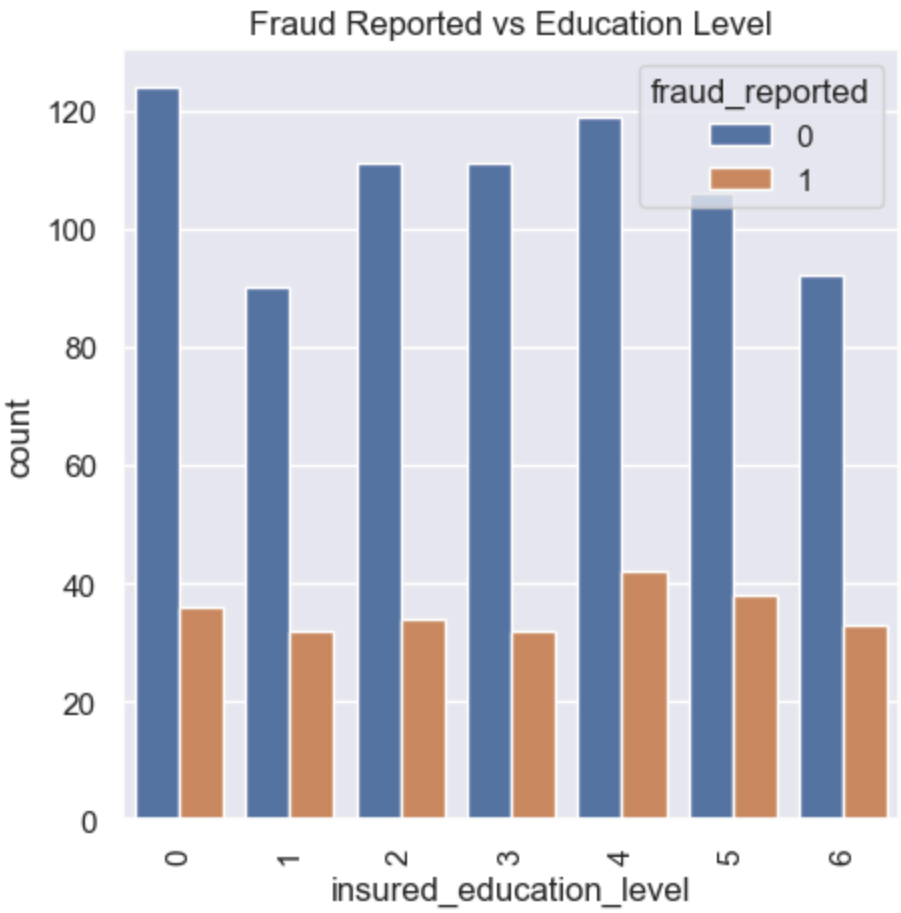


In [60]:
```
sns.countplot(x='incident_severity',hue='fraud_reported',data=df)
sns.set(rc={'figure.figsize':(5,5)})
plt.xticks(rotation=90)
plt.title("Fraud Reported vs Incident Severity")
```

Out[60]:
```
Text(0.5, 1.0, 'Fraud Reported vs Incident Severity')
```

In [61]:
```
sns.countplot(x='insured_education_level',hue='fraud_reported',data=df)
sns.set(rc={'figure.figsize':(5,5)})
plt.xticks(rotation=90)
plt.title("Fraud Reported vs Education Level")
```

Out[61]:
```
Text(0.5, 1.0, 'Fraud Reported vs Education Level')
```



In [62]:
```
sns.countplot(x='witnesses',hue='fraud_reported',data=df)
sns.set(rc={'figure.figsize':(5,5)})
plt.xticks(rotation=90)
plt.title("Fraud Reported vs Number of Witnesses")
```

Out[62]:
```
Text(0.5, 1.0, 'Fraud Reported vs Number of Witnesses')
```

In [63]:
```
#Using Binning to make the continuous data categorical
```

In [64]:
```
df["months_as_customer"] = pd.cut(df["months_as_customer"], bins=5, labels=[0, 1, 2
df["age"] = pd.cut(df["age"], bins=5, labels=[0, 1, 2, 3, 4])
df["policy_annual_premium"] = pd.cut(df["policy_annual_premium"], bins=5, labels=[0
df["capital-gains"] = pd.cut(df["capital-gains"], bins=5, labels=[0, 1, 2, 3, 4])
df["capital-loss"] = pd.cut(df["capital-loss"], bins=5, labels=[0, 1, 2, 3, 4])
df["total_claim_amount"] = pd.cut(df["total_claim_amount"], bins=5, labels=[0, 1, 2
df["injury_claim"] = pd.cut(df["injury_claim"], bins=5, labels=[0, 1, 2, 3, 4])
df["property_claim"] = pd.cut(df["property_claim"], bins=5, labels=[0, 1, 2, 3, 4])
df["vehicle_claim"] = pd.cut(df["vehicle_claim"], bins=5, labels=[0, 1, 2, 3, 4])
```

In [65]:
```
#Feature Scaling
```

In [66]:
```
min_max=MinMaxScaler()
numeric_columns=cols = ['policy_annual_premium', 'auto_year', 'bodily_injuries', 'a
```

In [67]:
```
df[numeric_columns] = min_max.fit_transform(df[numeric_columns])
```

In [68]:
```
print('Training Features Shape: ', df.shape)
```

```
Training Features Shape:  (1000, 31)
```

In [69]:
```
#Converting categorical data into numerical data
```

In [70]:
```
categorical_columns = df.select_dtypes(include=['object']).columns.tolist()
encoder = OneHotEncoder(sparse_output=False)
```

In [71]:
```
one_hot_encoded = encoder.fit_transform(df[categorical_columns])
```

In [72]:
```
one_hot_df = pd.DataFrame(one_hot_encoded, columns=encoder.get_feature_names_out(ca
```

In [73]:
```
df_encoded =pd.concat([df, one_hot_df], axis=1)
```

In [74]:
```
df=df_encoded.drop(categorical_columns, axis=1)
```

In [75]:
```
#Split the Data
```

In [76]:
```
X = df.iloc[:, 0:-1]
y = df.iloc[:, -1]
```

In [77]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [78]:
```
#Balance the Data
```

In [79]:
```
y_train.value_counts()
```

Out[79]:
```
0.0    744
1.0     56
Name: auto_make_Volkswagen, dtype: int64
```

In [80]:
```
sm=SMOTE()
X_train, y_train = sm.fit_resample(X_train, y_train)
```

In [81]:
```
y_train.value_counts()
```

Out[81]:
```
0.0    744
1.0    744
Name: auto_make_Volkswagen, dtype: int64
```

In [82]:
```
enn = EditedNearestNeighbours()
X_train, y_train = enn.fit_resample(X_train, y_train)
```

In [83]:
```
y_train.value_counts()
```

Out[83]:
```
0.0    744
1.0    744
Name: auto_make_Volkswagen, dtype: int64
```

In [84]:
```
#Develop the Models
```

In [85]:
```
# 1) Naive Bayes (Categorical)
```

In [85]:
```
cnb=CategoricalNB()
cnb.fit(X_train, y_train)
```

Out[85]:

```
▾ CategoricalNB
CategoricalNB()
```

In [86]:
```
y_pred_cnb=cnb.predict(X_test)
```

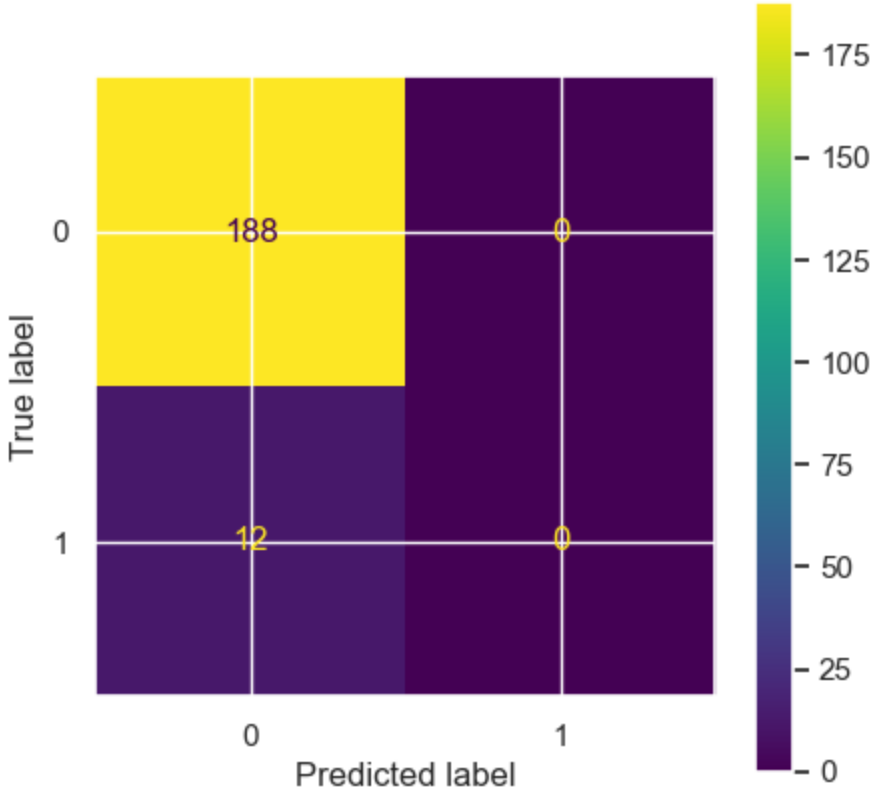
In [87]:
```
print(classification_report(y_test, y_pred_cnb))
```

```
              precision    recall  f1-score   support

         0.0       0.94      1.00      0.97       188
         1.0       0.00      0.00      0.00        12

    accuracy                           0.94       200
   macro avg       0.47      0.50      0.48       200
weighted avg       0.88      0.94      0.91       200
```

```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\carol\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\carol\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [88]:
```
acc_cnb=metrics.accuracy_score(y_test, y_pred_cnb)
```

In [89]:
```
labels = [0,1]
cm = confusion_matrix(y_test, y_pred_cnb, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```



In [90]:
```
#cross validation of CNB model
cross_val_results = cross_val_score(cnb, X_train, y_train, cv=10)
```

In [91]:
```
print(f'Cross-Validation Results (Accuracy): {cross_val_results}')
print(f'Mean Accuracy: {cross_val_results.mean()}')
```

```
Cross-Validation Results (Accuracy): [0.67785235 1.         0.98657718 1.         1.
1.
 1.         1.         1.         1.        ]
Mean Accuracy: 0.9664429530201343
```

In [93]:
```
cnb_mean_cv_acc=cross_val_results.mean()
```

In [140…
```
auc_cnb=metrics.roc_auc_score(y_test, y_pred_cnb)
print(auc_cnb)
```

```
0.5
```

In [94]:
```
# 2) SVC Model
```

In [95]:
```
svc=SVC()
svc.fit(X_train, y_train)
```

Out[95]:
```
SVC()
```

In [96]:
```
y_pred_svc=svc.predict(X_test)
```

In [97]:
```
print(classification_report(y_test, y_pred_svc))
```

```
              precision    recall  f1-score   support

         0.0       0.94      1.00      0.97       188
         1.0       0.00      0.00      0.00        12

    accuracy                           0.94       200
   macro avg       0.47      0.50      0.48       200
weighted avg       0.88      0.94      0.91       200
```

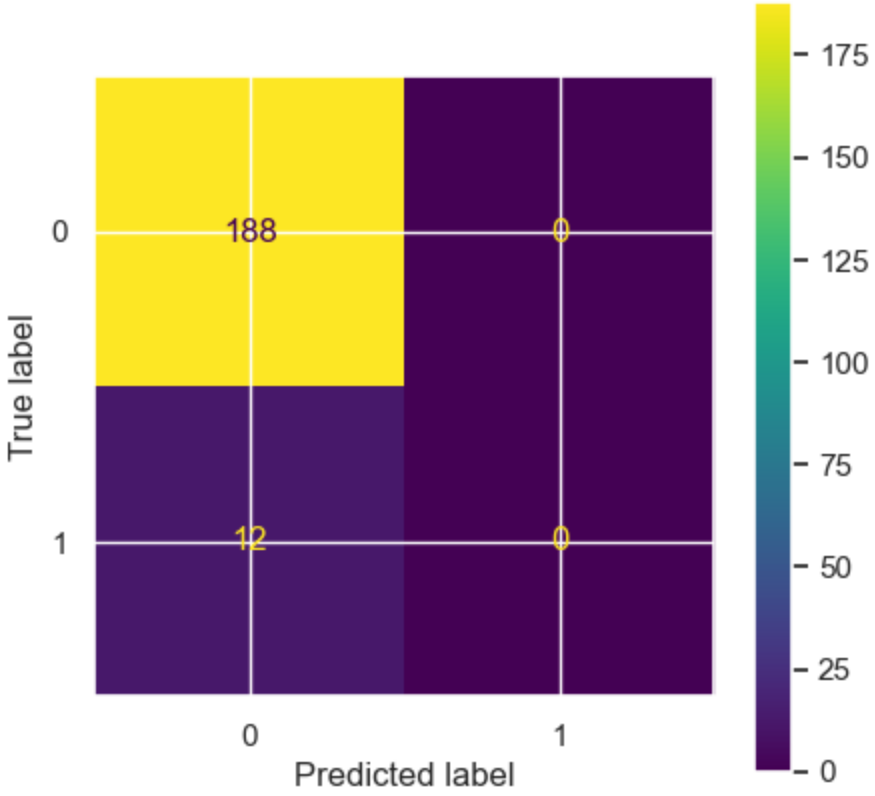
```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\carol\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\carol\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [98]:
```
cross_val_results = cross_val_score(svc, X_train, y_train, cv=10)
```

In [99]:
```
print(f'Cross-Validation Results (Accuracy): {cross_val_results}')
print(f'Mean Accuracy: {cross_val_results.mean()}')
```

```
Cross-Validation Results (Accuracy): [0.98657718 1.         1.         1.         1.
1.
 1.         1.         1.         1.        ]
Mean Accuracy: 0.9986577181208054
```

In [100…
```
labels = [0,1]
cm = confusion_matrix(y_test, y_pred_svc, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```



In [101…
```
#Hyperparameter Tuning for SVC
```

In [103…
```
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf', 'linear']}
grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)
grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.970 total time=   0.2s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.983 total time=   0.2s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.545 total time=   0.1s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.522 total time=   0.1s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.711 total time=   0.1s
[CV 1/5] END .....C=0.1, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END .....C=0.1, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END .....C=0.1, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END .....C=0.1, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END .....C=0.1, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.950 total time=   0.1s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.983 total time=   0.1s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.983 total time=   0.1s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.980 total time=   0.1s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.990 total time=   0.1s
[CV 1/5] END ...C=0.1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ...C=0.1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ...C=0.1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ...C=0.1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ...C=0.1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.555 total time=   0.1s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.518 total time=   0.1s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.572 total time=   0.1s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.552 total time=   0.1s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.748 total time=   0.1s
[CV 1/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.498 total time=   0.1s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.498 total time=   0.2s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.498 total time=   0.2s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.498 total time=   0.1s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.715 total time=   0.1s
[CV 1/5] END .C=0.1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END .C=0.1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END .C=0.1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END .C=0.1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END .C=0.1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.498 total time=   0.1s
[CV 2/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.498 total time=   0.1s
[CV 3/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.498 total time=   0.1s
[CV 4/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.498 total time=   0.1s
[CV 5/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.708 total time=   0.1s
[CV 1/5] END C=0.1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END C=0.1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END C=0.1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END C=0.1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END C=0.1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.990 total time=   0.1s
[CV 2/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.983 total time=   0.1s
[CV 3/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.983 total time=   0.1s
[CV 4/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.987 total time=   0.1s
[CV 5/5] END ..........C=1, gamma=1, kernel=rbf;, score=0.987 total time=   0.1s
```

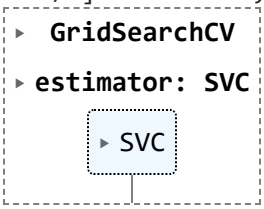
```
[CV 1/5] END .......C=1, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END .......C=1, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END .......C=1, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END .......C=1, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END .......C=1, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ........C=1, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END ........C=1, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 3/5] END ........C=1, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 4/5] END ........C=1, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END ........C=1, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 1/5] END .....C=1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END .....C=1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END .....C=1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END .....C=1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END .....C=1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.936 total time=   0.1s
[CV 2/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.943 total time=   0.1s
[CV 3/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.903 total time=   0.1s
[CV 4/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.950 total time=   0.1s
[CV 5/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.953 total time=   0.1s
[CV 1/5] END ....C=1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ....C=1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ....C=1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ....C=1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ....C=1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.612 total time=   0.1s
[CV 2/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.542 total time=   0.1s
[CV 3/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.619 total time=   0.1s
[CV 4/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.595 total time=   0.1s
[CV 5/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.715 total time=   0.1s
[CV 1/5] END ...C=1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ...C=1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ...C=1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ...C=1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ...C=1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.498 total time=   0.1s
[CV 2/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.498 total time=   0.1s
[CV 3/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.498 total time=   0.1s
[CV 4/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.498 total time=   0.1s
[CV 5/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.708 total time=   0.1s
[CV 1/5] END ..C=1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ..C=1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ..C=1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ..C=1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ..C=1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END .........C=10, gamma=1, kernel=rbf;, score=0.990 total time=   0.1s
[CV 2/5] END .........C=10, gamma=1, kernel=rbf;, score=0.983 total time=   0.1s
[CV 3/5] END .........C=10, gamma=1, kernel=rbf;, score=0.987 total time=   0.1s
[CV 4/5] END .........C=10, gamma=1, kernel=rbf;, score=0.987 total time=   0.1s
[CV 5/5] END .........C=10, gamma=1, kernel=rbf;, score=0.990 total time=   0.1s
[CV 1/5] END ......C=10, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ......C=10, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ......C=10, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ......C=10, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ......C=10, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
```

```
[CV 2/5] END .......C=10, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 3/5] END .......C=10, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 4/5] END .......C=10, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END .......C=10, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 1/5] END ....C=10, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ....C=10, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ....C=10, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ....C=10, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ....C=10, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 1/5] END ...C=10, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ...C=10, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ...C=10, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ...C=10, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ...C=10, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.903 total time=   0.1s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.890 total time=   0.1s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.823 total time=   0.1s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.910 total time=   0.1s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.903 total time=   0.1s
[CV 1/5] END ..C=10, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ..C=10, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ..C=10, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ..C=10, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ..C=10, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.615 total time=   0.1s
[CV 2/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.552 total time=   0.1s
[CV 3/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.629 total time=   0.1s
[CV 4/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.602 total time=   0.1s
[CV 5/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.708 total time=   0.1s
[CV 1/5] END .C=10, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END .C=10, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END .C=10, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END .C=10, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END .C=10, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ........C=100, gamma=1, kernel=rbf;, score=0.990 total time=   0.1s
[CV 2/5] END ........C=100, gamma=1, kernel=rbf;, score=0.983 total time=   0.1s
[CV 3/5] END ........C=100, gamma=1, kernel=rbf;, score=0.987 total time=   0.1s
[CV 4/5] END ........C=100, gamma=1, kernel=rbf;, score=0.987 total time=   0.1s
[CV 5/5] END ........C=100, gamma=1, kernel=rbf;, score=0.990 total time=   0.1s
[CV 1/5] END .....C=100, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END .....C=100, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END .....C=100, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END .....C=100, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END .....C=100, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ......C=100, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END ......C=100, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 3/5] END ......C=100, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 4/5] END ......C=100, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END ......C=100, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 1/5] END ...C=100, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ...C=100, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
```

```
[CV 3/5] END ...C=100, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ...C=100, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ...C=100, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END .....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END .....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 3/5] END .....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 4/5] END .....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END .....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 1/5] END ..C=100, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ..C=100, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ..C=100, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ..C=100, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ..C=100, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ....C=100, gamma=0.001, kernel=rbf;, score=1.000 total time=   0.1s
[CV 2/5] END ....C=100, gamma=0.001, kernel=rbf;, score=1.000 total time=   0.1s
[CV 3/5] END ....C=100, gamma=0.001, kernel=rbf;, score=1.000 total time=   0.1s
[CV 4/5] END ....C=100, gamma=0.001, kernel=rbf;, score=1.000 total time=   0.1s
[CV 5/5] END ....C=100, gamma=0.001, kernel=rbf;, score=1.000 total time=   0.1s
[CV 1/5] END .C=100, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END .C=100, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END .C=100, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END .C=100, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END .C=100, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.900 total time=   0.1s
[CV 2/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.876 total time=   0.1s
[CV 3/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.806 total time=   0.1s
[CV 4/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.893 total time=   0.1s
[CV 5/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.896 total time=   0.1s
[CV 1/5] END C=100, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END C=100, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END C=100, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END C=100, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END C=100, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.990 total time=   0.1s
[CV 2/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.983 total time=   0.1s
[CV 3/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.987 total time=   0.1s
[CV 4/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.987 total time=   0.1s
[CV 5/5] END .......C=1000, gamma=1, kernel=rbf;, score=0.990 total time=   0.1s
[CV 1/5] END ....C=1000, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ....C=1000, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ....C=1000, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ....C=1000, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ....C=1000, gamma=1, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 3/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 4/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 1/5] END ..C=1000, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END ..C=1000, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END ..C=1000, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END ..C=1000, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END ..C=1000, gamma=0.1, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 3/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
```

```
[CV 4/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 1/5] END .C=1000, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END .C=1000, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END .C=1000, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END .C=1000, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END .C=1000, gamma=0.01, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=1.000 total time=   0.1s
[CV 2/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=1.000 total time=   0.1s
[CV 3/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=1.000 total time=   0.1s
[CV 4/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=1.000 total time=   0.1s
[CV 5/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=1.000 total time=   0.1s
[CV 1/5] END C=1000, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END C=1000, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END C=1000, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END C=1000, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END C=1000, gamma=0.001, kernel=linear;, score=1.000 total time=   0.0s
[CV 1/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=1.000 total time=   0.5s
[CV 2/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=1.000 total time=   0.6s
[CV 3/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=1.000 total time=   0.5s
[CV 4/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=1.000 total time=   0.5s
[CV 5/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=1.000 total time=   0.5s
[CV 1/5] END C=1000, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 2/5] END C=1000, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 3/5] END C=1000, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 4/5] END C=1000, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
[CV 5/5] END C=1000, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.0s
```

Out[103…



In [104…

```
# print best parameter after tuning
print(grid.best_params_)
```

```
{'C': 0.1, 'gamma': 1, 'kernel': 'linear'}
```

In [105…

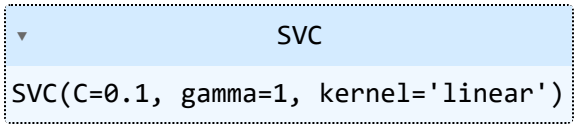
```
print(grid.best_estimator_)
```

```
SVC(C=0.1, gamma=1, kernel='linear')
```

In [102…

```
svc = SVC(C=0.1, gamma=1, kernel="linear")
svc.fit(X_train, y_train)
```

Out[102…



In [103…

```
y_pred_svc=svc.predict(X_test)
```

In [104…

```
print(classification_report(y_test, y_pred_svc))
```

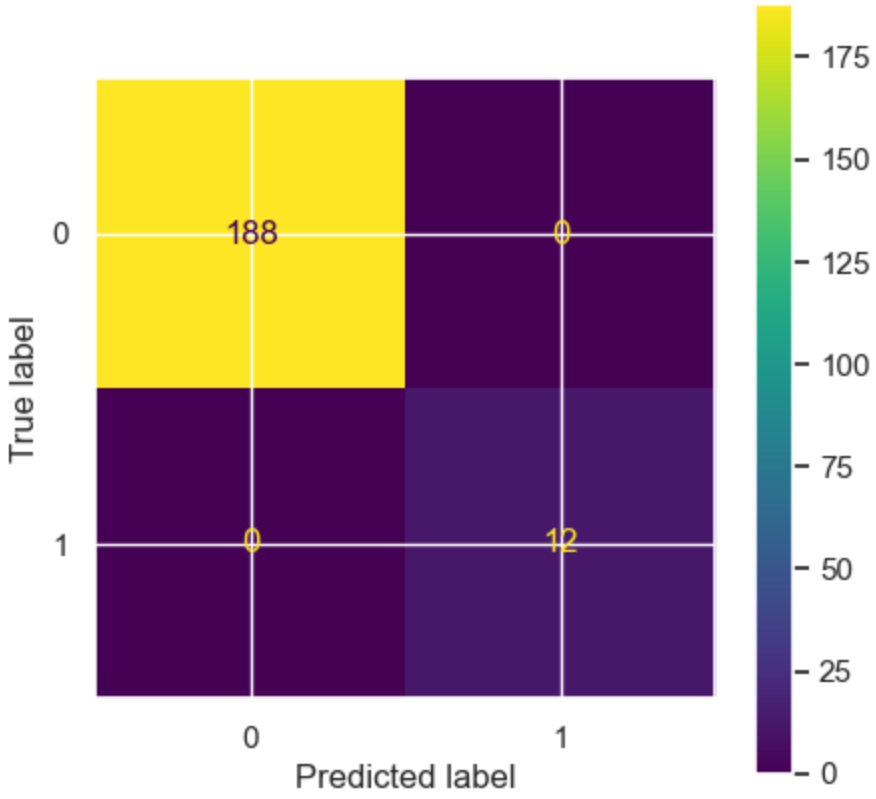
```
                   precision    recall  f1-score   support

              0.0       1.00      1.00      1.00       188
              1.0       1.00      1.00      1.00        12

         accuracy                           1.00       200
        macro avg       1.00      1.00      1.00       200
     weighted avg       1.00      1.00      1.00       200
```

In [105…
```
acc_svc=metrics.accuracy_score(y_test, y_pred_svc)
```

In [106…
```
auc_svc=metrics.roc_auc_score(y_test, y_pred_svc)
print(auc_svc)
```

```
1.0
```

In [107…
```
labels = [0,1]
cm = confusion_matrix(y_test, y_pred_svc, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```



In [108…
```
# 3) MLP Model
mlp = MLPClassifier()
mlp.fit(X_train, y_train)
```

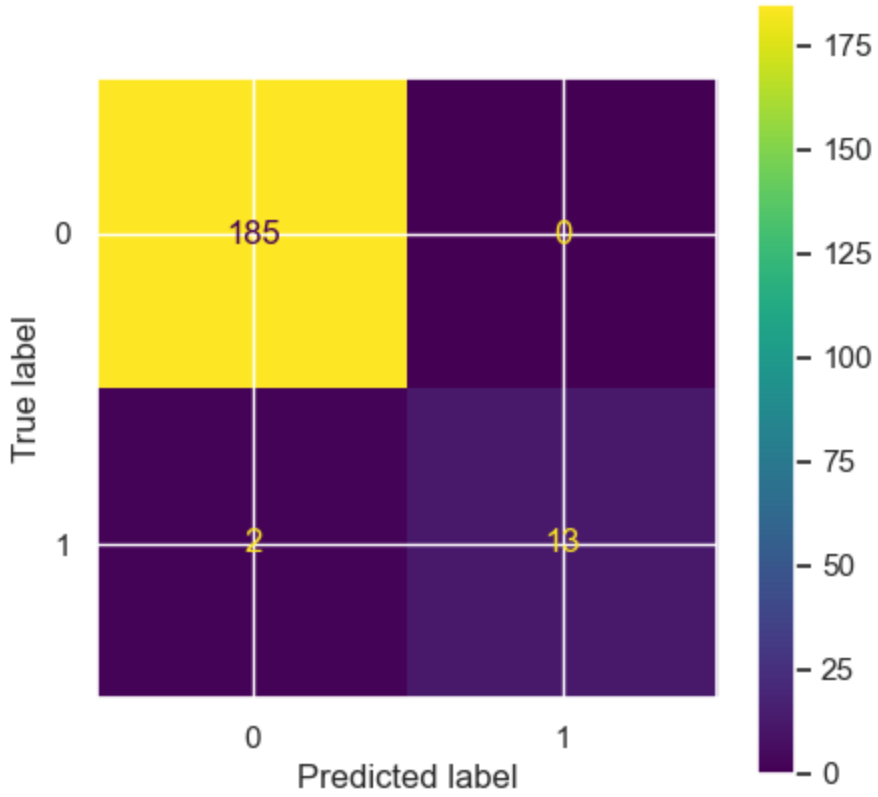
Out[108…
```
MLPClassifier()
```

In [113…
```
y_pred_mlp=mlp.predict(X_test)
```

In [114…
```
print(classification_report(y_test, y_pred_mlp))
```

```
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99       185
         1.0       1.00      0.87      0.93        15

    accuracy                           0.99       200
   macro avg       0.99      0.93      0.96       200
weighted avg       0.99      0.99      0.99       200
```

In [115…
```
labels = [0,1]
cm = confusion_matrix(y_test, y_pred_mlp, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```



In [116…
```
#Hyperparameter Tuning
param_grid = {
    'max_iter' :[500, 800],
    'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}
```

In [117…

```
grid = GridSearchCV(mlp, param_grid, refit = True, verbose = 3)
grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 96 candidates, totalling 480 fits
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   4.2s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   4.9s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   4.7s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   4.5s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   4.4s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=adam;, score=0.997 total time=   0.7s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.6s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   4.4s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.0s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   4.5s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   4.7s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   4.8s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   6.2s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   5.7s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   5.4s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   5.2s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   4.8s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.8s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.3s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.5s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.5s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.4s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.9s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   5.7s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   5.7s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   5.3s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   6.1s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   6.1s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.6s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.6s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
```

```
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.6s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   6.1s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.5s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   5.0s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   6.1s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.1s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=adam;, score=0.997 total time=   0.5s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   6.2s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   5.9s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   5.8s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   6.7s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   6.0s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.7s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.6s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.6s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   7.0s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.3s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.7s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.6s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.1s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=800, solver=adam;, score=0.997 total time=   0.7s
```

```
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.7s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.7s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=500, solver=sgd;, score=0.983 total time=   3.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=500, solver=sgd;, score=0.980 total time=   3.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=500, solver=sgd;, score=0.936 total time=   3.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=500, solver=sgd;, score=0.953 total time=   4.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=500, solver=sgd;, score=0.966 total time=   4.0s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.0s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.0s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.0s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=800, solver=sgd;, score=1.000 total time=   6.9s
```

```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   6.4s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.987 total time=   6.6s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.990 total time=   6.4s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   6.3s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.2s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.1s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=sgd;, score=0.977 total time=   5.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=sgd;, score=0.987 total time=   5.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=sgd;, score=0.936 total time=   7.7s
```

```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=sgd;, score=0.963 total time=   6.8s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=sgd;, score=0.970 total time=   6.1s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.8s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.7s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.7s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.4s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=  12.1s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=  10.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.980 total time=  10.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.997 total time=  10.8s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=800, solver=sgd;, score=0.997 total time=  11.8s
[CV 1/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.4s
[CV 2/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.2s
[CV 3/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.1s
[CV 4/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.2s
[CV 5/5] END activation=tanh, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   4.9s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=sgd;, score=0.997 total time=   5.3s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=sgd;, score=0.997 total time=   6.2s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   6.2s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   6.1s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.7s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   6.3s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.7s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   6.1s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.5s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   6.3s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.7s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.9s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.9s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.1s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   7.1s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   6.7s
```

```
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=sgd;, score=0.997 total time=   7.9s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   6.6s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   7.8s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.7s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.7s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.8s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.6s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   8.9s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   8.1s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   7.6s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.7s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.8s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.7s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.8s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   6.2s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   6.5s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   7.1s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   7.0s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   8.0s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.7s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.6s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.7s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.2s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.4s
```

```
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.9s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   6.4s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   6.2s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   6.0s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.8s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   2.3s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.2s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.4s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.4s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.5s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   8.7s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   8.2s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=sgd;, score=0.997 total time=   8.8s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   8.4s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   7.7s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   2.0s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.4s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.1s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.5s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.4s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.8s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   8.1s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   9.7s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   9.5s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.4s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.8s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.1s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.6s
```

```
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.3s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.2s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=sgd;, score=0.977 total time=   3.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=sgd;, score=0.977 total time=   4.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=sgd;, score=0.936 total time=   5.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=sgd;, score=0.953 total time=   5.6s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=sgd;, score=0.963 total time=   5.5s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=adam;, score=1.000 total time=   1.2s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=adam;, score=1.000 total time=   1.2s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=adam;, score=1.000 total time=   1.3s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=adam;, score=1.000 total time=   1.6s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=adam;, score=1.000 total time=   1.5s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=sgd;, score=1.000 total time=   7.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=sgd;, score=0.997 total time=   7.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=sgd;, score=0.987 total time=   6.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=sgd;, score=0.997 total time=   6.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=sgd;, score=1.000 total time=   6.2s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=adam;, score=1.000 total time=   1.2s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=adam;, score=1.000 total time=   1.1s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=adam;, score=1.000 total time=   1.4s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=adam;, score=1.000 total time=   1.1s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=adam;, score=1.000 total time=   1.1s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=sgd;, score=0.973 total time=   3.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=sgd;, score=0.967 total time=   4.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=sgd;, score=0.933 total time=   3.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=sgd;, score=0.943 total time=   5.4s
```

```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=sgd;, score=0.960 total time=   3.3s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.1s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.5s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.983 total time=   5.4s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.997 total time=   5.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.4s
[CV 1/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 2/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 3/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 4/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=800, solver=adam;, score=1.000 total time=   0.9s
[CV 5/5] END activation=tanh, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   5.0s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   4.2s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=sgd;, score=0.987 total time=   4.5s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=sgd;, score=0.997 total time=   4.8s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=sgd;, score=0.997 total time=   4.5s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.3s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.4s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=adam;, score=0.993 total time=   0.3s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.4s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=500, solver=adam;, score=0.997 total time=   0.4s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   4.6s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   5.5s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   4.6s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   4.3s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   4.6s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.4s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.3s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.3s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.4s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   4.6s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=sgd;, score=0.990 total time=   5.4s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learnin
g_rate=adaptive, max_iter=500, solver=sgd;, score=0.987 total time=   5.1s
```

```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   4.6s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   4.4s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.3s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.3s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.3s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.3s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.3s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.0s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.997 total time=   5.0s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.997 total time=   4.6s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.0s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   4.9s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.3s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=800, solver=adam;, score=0.993 total time=   0.3s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=800, solver=adam;, score=0.997 total time=   0.4s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.3s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 50, 50), learning_rate=adaptive, max_iter=800, solver=adam;, score=0.993 total time=   0.4s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=constant, max_iter=500, solver=sgd;, score=0.993 total time=   4.8s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=constant, max_iter=500, solver=sgd;, score=0.990 total time=   5.6s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=constant, max_iter=500, solver=sgd;, score=0.987 total time=   5.5s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=constant, max_iter=500, solver=sgd;, score=0.997 total time=   5.4s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=sgd;, score=0.997 total time=   5.8s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.4s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.4s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=adam;, score=0.997 total time=   0.4s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.4s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.4s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.3s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   5.5s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=sgd;, score=0.993 total time=   5.1s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.1s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   6.1s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=adam;, score=0.997 total time=   0.4s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.4s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=adam;, score=0.997 total time=   0.4s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=adam;, score=0.997 total time=   0.4s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.4s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   6.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   5.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=sgd;, score=0.990 total time=   6.7s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learni
ng_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   8.5s
```

```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=500, solver=sgd;, score=0.993 total time=   6.3s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.4s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=500, solver=adam;, score=0.993 total time=   0.4s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=500, solver=adam;, score=0.993 total time=   0.3s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.4s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=500, solver=adam;, score=0.997 total time=   0.4s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.993 total time=   4.9s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.3s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.6s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.993 total time=   6.6s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.0s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.3s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.4s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.4s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(50, 100, 50), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=sgd;, score=0.970 total time=   3.6s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=sgd;, score=0.980 total time=   4.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=sgd;, score=0.946 total time=   3.9s
```

```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=sgd;, score=0.957 total time=   3.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=sgd;, score=0.977 total time=   3.6s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.3s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.2s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.1s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   5.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   5.5s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.977 total time=   6.5s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.987 total time=   5.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=800, solver=sgd;, score=0.993 total time=   6.2s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.8s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.7s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=500, solver=sgd;, score=0.973 total time=   3.6s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=500, solver=sgd;, score=0.977 total time=   3.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=500, solver=sgd;, score=0.936 total time=   3.5s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=500, solver=sgd;, score=0.960 total time=   3.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=500, solver=sgd;, score=0.980 total time=   3.2s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate
=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.8s
```

```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.990 total time=   5.8s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.993 total time=   5.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.973 total time=   5.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.990 total time=   6.1s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.993 total time=   5.6s
[CV 1/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.7s
[CV 2/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.7s
[CV 3/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.7s
[CV 4/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.9s
[CV 5/5] END activation=relu, alpha=0.0001, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.9s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_rate=constant, max_iter=500, solver=sgd;, score=0.997 total time=   5.2s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=sgd;, score=0.993 total time=   5.4s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=sgd;, score=0.993 total time=   5.1s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=sgd;, score=0.993 total time=   5.5s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   4.1s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.6s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=adam;, score=0.997 total time=   0.5s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.5s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.0s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   5.3s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=sgd;, score=0.983 total time=   4.4s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   5.3s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   4.8s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=sgd;, score=0.993 total time=   4.5s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=sgd;, score=0.993 total time=   4.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=sgd;, score=0.993 total time=   4.9s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=sgd;, score=0.997 total time=   5.2s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=sgd;, score=0.997 total time=   7.8s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.7s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.7s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=adam;, score=0.997 total time=   0.7s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.7s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   0.6s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=sgd;, score=0.997 total time=   5.2s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=sgd;, score=0.993 total time=   5.9s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=sgd;, score=0.987 total time=   4.7s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=sgd;, score=0.997 total time=   5.0s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   5.2s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.6s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 50, 50), learning_
rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   0.5s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   5.9s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=sgd;, score=0.990 total time=   5.1s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=sgd;, score=0.987 total time=   5.3s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=sgd;, score=0.997 total time=   6.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=sgd;, score=1.000 total time=   7.4s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.4s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.0s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.1s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.1s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.3s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=sgd;, score=0.997 total time=   5.5s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.1s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=sgd;, score=0.993 total time=   6.9s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=sgd;, score=0.987 total time=   7.2s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=sgd;, score=1.000 total time=   5.3s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.3s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=constant, max_iter=800, solver=adam;, score=1.000 total time=   0.8s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   5.8s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   5.8s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=sgd;, score=0.993 total time=   6.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
```

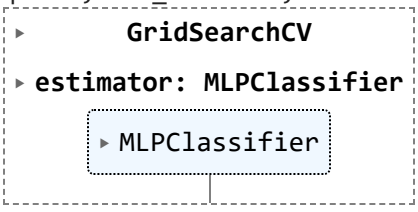
```
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=sgd;, score=0.997 total time=   6.3s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=sgd;, score=1.000 total time=   6.8s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.0s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.1s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.0s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=adam;, score=1.000 total time=   1.2s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=500, solver=adam;, score=0.997 total time=   0.9s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=sgd;, score=0.997 total time=   5.5s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=sgd;, score=0.997 total time=   9.4s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=sgd;, score=0.993 total time=   8.8s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   6.4s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=sgd;, score=1.000 total time=   7.2s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.2s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.3s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.7s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.4s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(50, 100, 50), learning
_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.2s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=sgd;, score=0.963 total time=   4.5s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=sgd;, score=0.963 total time=   4.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=500, solver=sgd;, score=0.943 total time=   3.7s
```

```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=sgd;, score=0.960 total time=   5.4s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=sgd;, score=0.977 total time=   4.8s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.1s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.0s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   1.0s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.990 total time=   5.5s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.993 total time=   5.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.973 total time=   5.8s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=constant, max_iter=800, solver=sgd;, score=0.993 total time=   6.4s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=sgd;, score=1.000 total time=   6.0s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=adam;, score=1.000 total time=   0.9s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=c
onstant, max_iter=800, solver=adam;, score=1.000 total time=   0.9s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=sgd;, score=0.963 total time=   3.6s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=sgd;, score=0.970 total time=   4.1s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=sgd;, score=0.926 total time=   3.7s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=sgd;, score=0.963 total time=   3.6s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=sgd;, score=0.980 total time=   4.4s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=adam;, score=1.000 total time=   1.0s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=adam;, score=1.000 total time=   0.9s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=adam;, score=1.000 total time=   1.9s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=adam;, score=1.000 total time=   1.1s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=a
daptive, max_iter=500, solver=adam;, score=1.000 total time=   1.2s
```

```
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.990 total time=   6.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.990 total time=   6.2s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.983 total time=   7.0s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.993 total time=   8.3s
C:\Users\carol\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (800) reached and the optimization hasn't converged yet.
  warnings.warn(
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=sgd;, score=0.997 total time=   7.8s
[CV 1/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 2/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.1s
[CV 3/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.4s
[CV 4/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
[CV 5/5] END activation=relu, alpha=0.05, hidden_layer_sizes=(100,), learning_rate=adaptive, max_iter=800, solver=adam;, score=1.000 total time=   1.0s
```

Out[117…



In [118…

```
print(grid.best_params_)
```

```
{'activation': 'tanh', 'alpha': 0.0001, 'hidden_layer_sizes': (50, 50, 50), 'learning_rate': 'constant', 'max_iter': 500, 'solver': 'sgd'}
```

In [109…

```
mlp = MLPClassifier(max_iter=500, activation="tanh", alpha=0.0001, hidden_layer_siz
mlp.fit(X_train, y_train)
```

Out[109…

MLPClassifier

```
MLPClassifier(activation='tanh', hidden_layer_sizes=(50, 50, 50), max_iter
=500,
              solver='sgd')
```

In [110…
```
y_pred_mlp=mlp.predict(X_test)
```

In [111…
```
print(classification_report(y_test, y_pred_mlp))
```

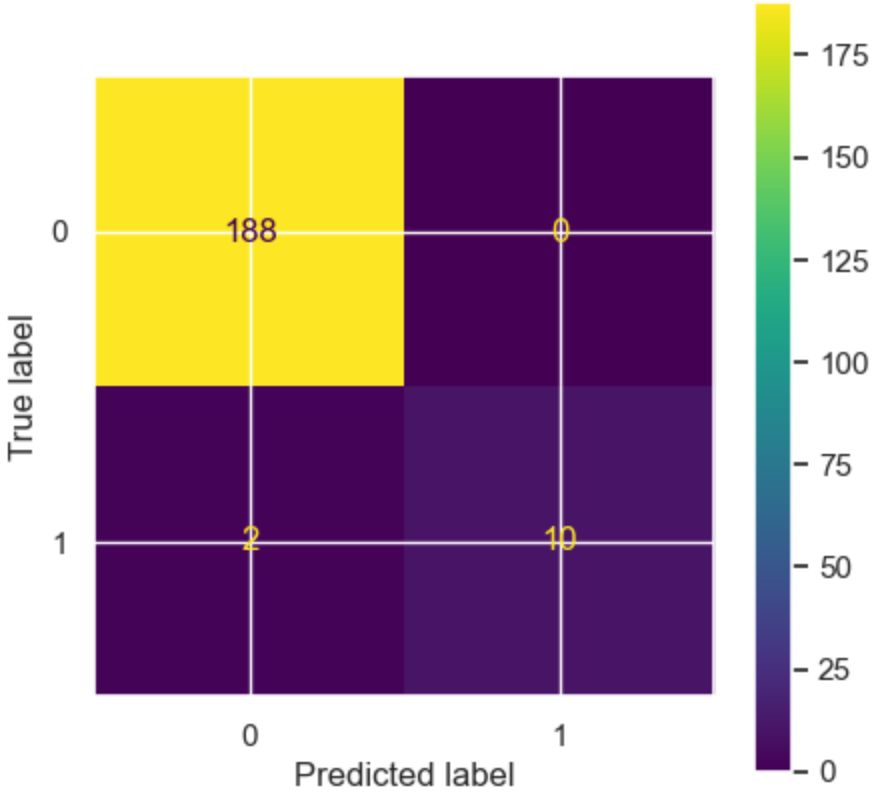
```
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99       188
         1.0       1.00      0.83      0.91        12

    accuracy                           0.99       200
   macro avg       0.99      0.92      0.95       200
weighted avg       0.99      0.99      0.99       200
```

In [112…
```
acc_mlp=metrics.accuracy_score(y_test, y_pred_mlp)
```

In [113…
```
auc_mlp=metrics.roc_auc_score(y_test, y_pred_mlp)
print(auc_mlp)
```

```
0.9166666666666667
```

In [114…
```
labels = [0,1]
cm = confusion_matrix(y_test, y_pred_mlp, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```



Type text here

In [115…
```
# 4) Decision Tree
```

In [126…
```
dt = DecisionTreeClassifier()
dt = dt.fit(X_train, y_train)
```

In [127…
```
plt.figure(figsize=(12,12))
```

Out[127… <Figure size 1200x1200 with 0 Axes>

<Figure size 1200x1200 with 0 Axes>

In [128…
```
tree.plot_tree(dt)
```

Out[128… [Text(0.5425237341772152, 0.9772727272727273, 'x[8] <= 0.0\ngini = 0.5\nsamples = 1494\nvalue = [747, 747]'),
 Text(0.2986550632911392, 0.9318181818181818, 'x[38] <= 0.008\ngini = 0.322\nsamples = 480\nvalue = [383, 97]'),
 Text(0.21123417721518986, 0.8863636363636364, 'x[3] <= 0.503\ngini = 0.251\nsamples = 428\nvalue = [365, 63]'),
 Text(0.1360759493670886, 0.8409090909090909, 'x[7] <= 0.776\ngini = 0.103\nsamples = 275\nvalue = [260, 15]'),
 Text(0.0949367088607595, 0.7954545454545454, 'x[44] <= 0.5\ngini = 0.073\nsamples = 264\nvalue = [254, 10]'),
 Text(0.05063291139240506, 0.75, 'x[34] <= 0.216\ngini = 0.04\nsamples = 248\nvalue = [243, 5]'),
 Text(0.02531645569620253, 0.7045454545454546, 'x[18] <= 0.975\ngini = 0.017\nsamples = 231\nvalue = [229, 2]'),
 Text(0.012658227848101266, 0.6590909090909091, 'gini = 0.0\nsamples = 222\nvalue = [222, 0]'),
 Text(0.0379746835443038, 0.6590909090909091, 'x[4] <= 0.318\ngini = 0.346\nsamples = 9\nvalue = [7, 2]'),
 Text(0.02531645569620253, 0.6136363636363636, 'gini = 0.0\nsamples = 6\nvalue = [6, 0]'),
 Text(0.05063291139240506, 0.6136363636363636, 'x[12] <= 0.5\ngini = 0.444\nsamples = 3\nvalue = [1, 2]'),
 Text(0.0379746835443038, 0.5681818181818182, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.06329113924050633, 0.5681818181818182, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.0759493670886076, 0.7045454545454546, 'x[6] <= 0.131\ngini = 0.291\nsamples = 17\nvalue = [14, 3]'),
 Text(0.06329113924050633, 0.6590909090909091, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.08860759493670886, 0.6590909090909091, 'x[53] <= 0.5\ngini = 0.124\nsamples = 15\nvalue = [14, 1]'),
 Text(0.0759493670886076, 0.6136363636363636, 'gini = 0.0\nsamples = 14\nvalue = [14, 0]'),
 Text(0.10126582278481013, 0.6136363636363636, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.13924050632911392, 0.75, 'x[28] <= 0.279\ngini = 0.43\nsamples = 16\nvalue = [11, 5]'),
 Text(0.12658227848101267, 0.7045454545454546, 'x[10] <= 0.087\ngini = 0.153\nsamples = 12\nvalue = [11, 1]'),
 Text(0.11392405063291139, 0.6590909090909091, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.13924050632911392, 0.6590909090909091, 'gini = 0.0\nsamples = 11\nvalue = [11, 0]'),
 Text(0.1518987341772152, 0.7045454545454546, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.17721518987341772, 0.7954545454545454, 'x[68] <= 0.5\ngini = 0.496\nsamples = 11\nvalue = [6, 5]'),
 Text(0.16455696202531644, 0.75, 'gini = 0.0\nsamples = 5\nvalue = [5, 0]'),
 Text(0.189873417721519, 0.75, 'x[0] <= 0.4\ngini = 0.278\nsamples = 6\nvalue = [1, 5]'),
 Text(0.17721518987341772, 0.7045454545454546, 'gini = 0.0\nsamples = 5\nvalue = [0, 5]'),
 Text(0.20253164556962025, 0.7045454545454546, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.28639240506329117, 0.8409090909090909, 'x[3] <= 0.749\ngini = 0.431\nsampl

```
es = 153\nvalue = [105, 48]'),
 Text(0.2737341772151899, 0.7954545454545454, 'gini = 0.0\nsamples = 21\nvalue =
[0, 21]'),
 Text(0.2990506329113924, 0.7954545454545454, 'x[48] <= 0.043\ngini = 0.325\nsampl
es = 132\nvalue = [105, 27]'),
 Text(0.2626582278481013, 0.75, 'x[50] <= 0.077\ngini = 0.15\nsamples = 110\nvalue
= [101, 9]'),
 Text(0.22784810126582278, 0.7045454545454546, 'x[10] <= 0.978\ngini = 0.092\nsamp
les = 104\nvalue = [99, 5]'),
 Text(0.1962025316455696, 0.6590909090909091, 'x[60] <= 0.5\ngini = 0.059\nsamples
= 98\nvalue = [95, 3]'),
 Text(0.17088607594936708, 0.6136363636363636, 'x[9] <= 0.167\ngini = 0.023\nsampl
es = 85\nvalue = [84, 1]'),
 Text(0.15822784810126583, 0.5681818181818182, 'x[80] <= 0.5\ngini = 0.219\nsample
s = 8\nvalue = [7, 1]'),
 Text(0.14556962025316456, 0.5227272727272727, 'gini = 0.0\nsamples = 7\nvalue =
[7, 0]'),
 Text(0.17088607594936708, 0.5227272727272727, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.18354430379746836, 0.5681818181818182, 'gini = 0.0\nsamples = 77\nvalue =
[77, 0]'),
 Text(0.22151898734177214, 0.6136363636363636, 'x[9] <= 0.833\ngini = 0.26\nsample
s = 13\nvalue = [11, 2]'),
 Text(0.2088607594936709, 0.5681818181818182, 'gini = 0.0\nsamples = 10\nvalue =
[10, 0]'),
 Text(0.23417721518987342, 0.5681818181818182, 'x[30] <= 0.5\ngini = 0.444\nsample
s = 3\nvalue = [1, 2]'),
 Text(0.22151898734177214, 0.5227272727272727, 'gini = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(0.2468354430379747, 0.5227272727272727, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.25949367088607594, 0.6590909090909091, 'x[78] <= 0.5\ngini = 0.444\nsample
s = 6\nvalue = [4, 2]'),
 Text(0.2468354430379747, 0.6136363636363636, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(0.2721518987341772, 0.6136363636363636, 'x[55] <= 0.5\ngini = 0.444\nsamples
= 3\nvalue = [1, 2]'),
 Text(0.25949367088607594, 0.5681818181818182, 'gini = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(0.2848101265822785, 0.5681818181818182, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.2974683544303797, 0.7045454545454546, 'x[76] <= 0.5\ngini = 0.444\nsamples
= 6\nvalue = [2, 4]'),
 Text(0.2848101265822785, 0.6590909090909091, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.310126582278481, 0.6590909090909091, 'gini = 0.0\nsamples = 4\nvalue = [0,
4]'),
 Text(0.33544303797468356, 0.75, 'x[13] <= 0.652\ngini = 0.298\nsamples = 22\nvalu
e = [4, 18]'),
 Text(0.3227848101265823, 0.7045454545454546, 'gini = 0.0\nsamples = 17\nvalue =
[0, 17]'),
 Text(0.34810126582278483, 0.7045454545454546, 'x[65] <= 0.365\ngini = 0.32\nsampl
es = 5\nvalue = [4, 1]'),
 Text(0.33544303797468356, 0.6590909090909091, 'gini = 0.0\nsamples = 4\nvalue =
[4, 0]'),
 Text(0.36075949367088606, 0.6590909090909091, 'gini = 0.0\nsamples = 1\nvalue =
```

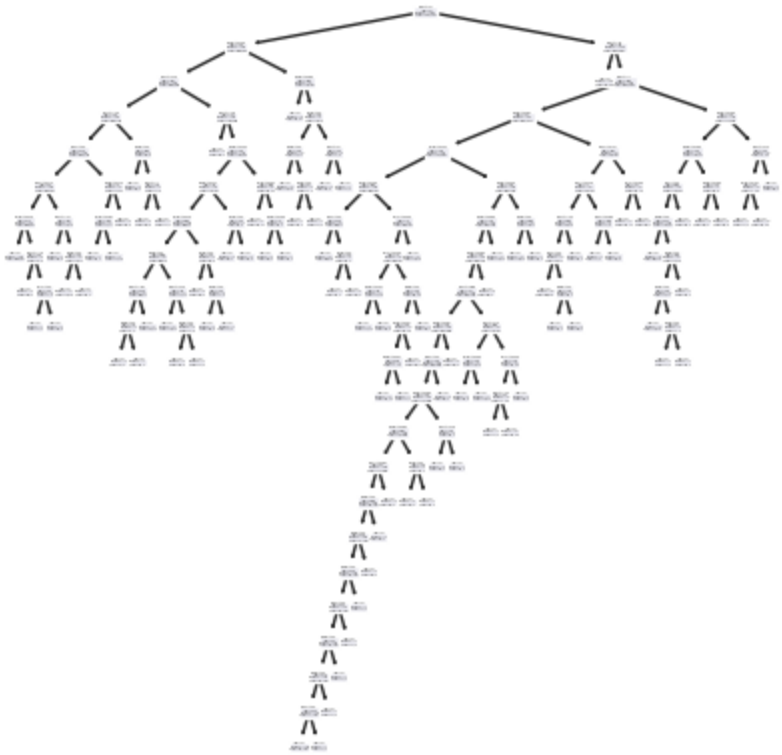
```
[0, 1]'),
 Text(0.3860759493670886, 0.8863636363636364, 'x[38] <= 0.998\ngini = 0.453\nsampl
es = 52\nvalue = [18, 34]'),
 Text(0.37341772151898733, 0.8409090909090909, 'gini = 0.0\nsamples = 31\nvalue =
[0, 31]'),
 Text(0.3987341772151899, 0.8409090909090909, 'x[73] <= 0.5\ngini = 0.245\nsamples
= 21\nvalue = [18, 3]'),
 Text(0.37341772151898733, 0.7954545454545454, 'x[58] <= 0.5\ngini = 0.105\nsample
s = 18\nvalue = [17, 1]'),
 Text(0.36075949367088606, 0.75, 'gini = 0.0\nsamples = 16\nvalue = [16, 0]'),
 Text(0.3860759493670886, 0.75, 'x[65] <= 0.5\ngini = 0.5\nsamples = 2\nvalue =
[1, 1]'),
 Text(0.37341772151898733, 0.7045454545454546, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.3987341772151899, 0.7045454545454546, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.4240506329113924, 0.7954545454545454, 'x[5] <= 0.5\ngini = 0.444\nsamples
= 3\nvalue = [1, 2]'),
 Text(0.41139240506329117, 0.75, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.43670886075949367, 0.75, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.7863924050632911, 0.9318181818181818, 'x[8] <= 0.25\ngini = 0.46\nsamples
= 1014\nvalue = [364, 650]'),
 Text(0.7737341772151899, 0.8863636363636364, 'gini = 0.0\nsamples = 218\nvalue =
[0, 218]'),
 Text(0.7990506329113924, 0.8863636363636364, 'x[9] <= 1.0\ngini = 0.496\nsamples
= 796\nvalue = [364, 432]'),
 Text(0.6677215189873418, 0.8409090909090909, 'x[68] <= 0.997\ngini = 0.471\nsampl
es = 661\nvalue = [251, 410]'),
 Text(0.5569620253164557, 0.7954545454545454, 'x[13] <= 0.334\ngini = 0.429\nsampl
es = 562\nvalue = [175, 387]'),
 Text(0.46835443037974683, 0.75, 'x[76] <= 0.009\ngini = 0.493\nsamples = 166\nval
ue = [93, 73]'),
 Text(0.4240506329113924, 0.7045454545454546, 'x[42] <= 0.5\ngini = 0.033\nsamples
= 59\nvalue = [58, 1]'),
 Text(0.41139240506329117, 0.6590909090909091, 'gini = 0.0\nsamples = 56\nvalue =
[56, 0]'),
 Text(0.43670886075949367, 0.6590909090909091, 'x[78] <= 0.5\ngini = 0.444\nsample
s = 3\nvalue = [2, 1]'),
 Text(0.4240506329113924, 0.6136363636363636, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.44936708860759494, 0.6136363636363636, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.5126582278481012, 0.7045454545454546, 'x[12] <= 0.968\ngini = 0.44\nsample
s = 107\nvalue = [35, 72]'),
 Text(0.5, 0.6590909090909091, 'x[12] <= 0.009\ngini = 0.34\nsamples = 92\nvalue =
[20, 72]'),
 Text(0.47468354430379744, 0.6136363636363636, 'x[45] <= 0.047\ngini = 0.26\nsampl
es = 13\nvalue = [11, 2]'),
 Text(0.4620253164556962, 0.5681818181818182, 'gini = 0.0\nsamples = 11\nvalue =
[11, 0]'),
 Text(0.4873417721518987, 0.5681818181818182, 'gini = 0.0\nsamples = 2\nvalue =
[0, 2]'),
 Text(0.5253164556962026, 0.6136363636363636, 'x[93] <= 0.5\ngini = 0.202\nsamples
= 79\nvalue = [9, 70]'),
 Text(0.5126582278481012, 0.5681818181818182, 'x[73] <= 0.601\ngini = 0.102\nsampl
es = 74\nvalue = [4, 70]'),
```

```
 Text(0.5, 0.5227272727272727, 'x[60] <= 0.937\ngini = 0.028\nsamples = 71\nvalue
= [1, 70]'),
 Text(0.4873417721518987, 0.4772727272727273, 'gini = 0.0\nsamples = 70\nvalue =
[0, 70]'),
 Text(0.5126582278481012, 0.4772727272727273, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.5253164556962026, 0.5227272727272727, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(0.5379746835443038, 0.5681818181818182, 'gini = 0.0\nsamples = 5\nvalue =
[5, 0]'),
 Text(0.5253164556962026, 0.6590909090909091, 'gini = 0.0\nsamples = 15\nvalue =
[15, 0]'),
 Text(0.6455696202531646, 0.75, 'x[58] <= 0.965\ngini = 0.328\nsamples = 396\nvalu
e = [82, 314]'),
 Text(0.620253164556962, 0.7045454545454546, 'x[65] <= 0.898\ngini = 0.279\nsample
s = 376\nvalue = [63, 313]'),
 Text(0.6075949367088608, 0.6590909090909091, 'x[60] <= 0.999\ngini = 0.234\nsampl
es = 362\nvalue = [49, 313]'),
 Text(0.5949367088607594, 0.6136363636363636, 'x[2] <= 0.999\ngini = 0.201\nsample
s = 353\nvalue = [40, 313]'),
 Text(0.5632911392405063, 0.5681818181818182, 'x[28] <= 0.981\ngini = 0.139\nsampl
es = 319\nvalue = [24, 295]'),
 Text(0.5506329113924051, 0.5227272727272727, 'x[92] <= 0.5\ngini = 0.119\nsamples
= 315\nvalue = [20, 295]'),
 Text(0.5379746835443038, 0.4772727272727273, 'x[40] <= 0.995\ngini = 0.103\nsampl
es = 312\nvalue = [17, 295]'),
 Text(0.5063291139240507, 0.4318181818181818, 'x[39] <= 0.993\ngini = 0.087\nsampl
es = 308\nvalue = [14, 294]'),
 Text(0.4810126582278481, 0.38636363636363635, 'x[41] <= 0.997\ngini = 0.07\nsampl
es = 304\nvalue = [11, 293]'),
 Text(0.46835443037974683, 0.3409090909090909, 'x[88] <= 0.5\ngini = 0.058\nsample
s = 302\nvalue = [9, 293]'),
 Text(0.45569620253164556, 0.29545454545454547, 'x[83] <= 0.5\ngini = 0.046\nsampl
es = 300\nvalue = [7, 293]'),
 Text(0.4430379746835443, 0.25, 'x[89] <= 0.5\ngini = 0.033\nsamples = 298\nvalue
= [5, 293]'),
 Text(0.43037974683544306, 0.20454545454545456, 'x[82] <= 0.5\ngini = 0.027\nsampl
es = 297\nvalue = [4, 293]'),
 Text(0.4177215189873418, 0.1590909090909091, 'x[90] <= 0.5\ngini = 0.02\nsamples
= 296\nvalue = [3, 293]'),
 Text(0.4050632911392405, 0.11363636363636363, 'x[85] <= 0.5\ngini = 0.013\nsample
s = 295\nvalue = [2, 293]'),
 Text(0.3924050632911392, 0.06818181818181818, 'x[52] <= 0.5\ngini = 0.007\nsample
s = 294\nvalue = [1, 293]'),
 Text(0.379746835443038, 0.022727272727272728, 'gini = 0.0\nsamples = 293\nvalue =
[0, 293]'),
 Text(0.4050632911392405, 0.022727272727272728, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.4177215189873418, 0.06818181818181818, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.43037974683544306, 0.11363636363636363, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.4430379746835443, 0.1590909090909091, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.45569620253164556, 0.20454545454545456, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
```

```
 Text(0.46835443037974683, 0.25, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.4810126582278481, 0.29545454545454547, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.4936708860759494, 0.3409090909090909, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.5316455696202531, 0.38636363636363635, 'x[78] <= 0.5\ngini = 0.375\nsample
s = 4\nvalue = [3, 1]'),
 Text(0.5189873417721519, 0.3409090909090909, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(0.5443037974683544, 0.3409090909090909, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.569620253164557, 0.4318181818181818, 'x[0] <= 0.875\ngini = 0.375\nsamples
= 4\nvalue = [3, 1]'),
 Text(0.5569620253164557, 0.38636363636363635, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(0.5822784810126582, 0.38636363636363635, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.5632911392405063, 0.4772727272727273, 'gini = 0.0\nsamples = 3\nvalue =
[3, 0]'),
 Text(0.5759493670886076, 0.5227272727272727, 'gini = 0.0\nsamples = 4\nvalue =
[4, 0]'),
 Text(0.6265822784810127, 0.5681818181818182, 'x[6] <= 0.673\ngini = 0.498\nsample
s = 34\nvalue = [16, 18]'),
 Text(0.6012658227848101, 0.5227272727272727, 'x[13] <= 0.608\ngini = 0.305\nsampl
es = 16\nvalue = [13, 3]'),
 Text(0.5886075949367089, 0.4772727272727273, 'gini = 0.0\nsamples = 3\nvalue =
[0, 3]'),
 Text(0.6139240506329114, 0.4772727272727273, 'gini = 0.0\nsamples = 13\nvalue =
[13, 0]'),
 Text(0.6518987341772152, 0.5227272727272727, 'x[14] <= 0.625\ngini = 0.278\nsampl
es = 18\nvalue = [3, 15]'),
 Text(0.6392405063291139, 0.4772727272727273, 'x[9] <= 0.114\ngini = 0.117\nsample
s = 16\nvalue = [1, 15]'),
 Text(0.6265822784810127, 0.4318181818181818, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.6518987341772152, 0.4318181818181818, 'gini = 0.0\nsamples = 15\nvalue =
[0, 15]'),
 Text(0.6645569620253164, 0.4772727272727273, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.620253164556962, 0.6136363636363636, 'gini = 0.0\nsamples = 9\nvalue = [9,
0]'),
 Text(0.6329113924050633, 0.6590909090909091, 'gini = 0.0\nsamples = 14\nvalue =
[14, 0]'),
 Text(0.6708860759493671, 0.7045454545454546, 'x[55] <= 0.5\ngini = 0.095\nsamples
= 20\nvalue = [19, 1]'),
 Text(0.6582278481012658, 0.6590909090909091, 'gini = 0.0\nsamples = 19\nvalue =
[19, 0]'),
 Text(0.6835443037974683, 0.6590909090909091, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.7784810126582279, 0.7954545454545454, 'x[47] <= 0.121\ngini = 0.357\nsampl
es = 99\nvalue = [76, 23]'),
 Text(0.7468354430379747, 0.75, 'x[41] <= 0.079\ngini = 0.21\nsamples = 84\nvalue
= [74, 10]'),
 Text(0.7215189873417721, 0.7045454545454546, 'x[4] <= 0.773\ngini = 0.052\nsample
s = 75\nvalue = [73, 2]'),
 Text(0.7088607594936709, 0.6590909090909091, 'x[22] <= 0.5\ngini = 0.027\nsamples
```

```
= 74\nvalue = [73, 1]'),
 Text(0.6962025316455697, 0.6136363636363636, 'gini = 0.0\nsamples = 68\nvalue =
[68, 0]'),
 Text(0.7215189873417721, 0.6136363636363636, 'x[80] <= 0.5\ngini = 0.278\nsamples
= 6\nvalue = [5, 1]'),
 Text(0.7088607594936709, 0.5681818181818182, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.7341772151898734, 0.5681818181818182, 'gini = 0.0\nsamples = 5\nvalue =
[5, 0]'),
 Text(0.7341772151898734, 0.6590909090909091, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.7721518987341772, 0.7045454545454546, 'x[57] <= 0.088\ngini = 0.198\nsampl
es = 9\nvalue = [1, 8]'),
 Text(0.759493670886076, 0.6590909090909091, 'gini = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.7848101265822784, 0.6590909090909091, 'gini = 0.0\nsamples = 8\nvalue =
[0, 8]'),
 Text(0.810126582278481, 0.75, 'x[12] <= 0.459\ngini = 0.231\nsamples = 15\nvalue
= [2, 13]'),
 Text(0.7974683544303798, 0.7045454545454546, 'gini = 0.0\nsamples = 13\nvalue =
[0, 13]'),
 Text(0.8227848101265823, 0.7045454545454546, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.930379746835443, 0.8409090909090909, 'x[35] <= 0.298\ngini = 0.273\nsample
s = 135\nvalue = [113, 22]'),
 Text(0.8860759493670886, 0.7954545454545454, 'x[50] <= 0.298\ngini = 0.116\nsampl
es = 113\nvalue = [106, 7]'),
 Text(0.8607594936708861, 0.75, 'x[21] <= 0.5\ngini = 0.055\nsamples = 107\nvalue
= [104, 3]'),
 Text(0.8481012658227848, 0.7045454545454546, 'x[59] <= 0.5\ngini = 0.037\nsamples
= 106\nvalue = [104, 2]'),
 Text(0.8354430379746836, 0.6590909090909091, 'gini = 0.0\nsamples = 87\nvalue =
[87, 0]'),
 Text(0.8607594936708861, 0.6590909090909091, 'x[26] <= 0.5\ngini = 0.188\nsamples
= 19\nvalue = [17, 2]'),
 Text(0.8481012658227848, 0.6136363636363636, 'x[33] <= 0.5\ngini = 0.105\nsamples
= 18\nvalue = [17, 1]'),
 Text(0.8354430379746836, 0.5681818181818182, 'gini = 0.0\nsamples = 16\nvalue =
[16, 0]'),
 Text(0.8607594936708861, 0.5681818181818182, 'x[68] <= 0.5\ngini = 0.5\nsamples =
2\nvalue = [1, 1]'),
 Text(0.8481012658227848, 0.5227272727272727, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.8734177215189873, 0.5227272727272727, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.8734177215189873, 0.6136363636363636, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.8734177215189873, 0.7045454545454546, 'gini = 0.0\nsamples = 1\nvalue =
[0, 1]'),
 Text(0.9113924050632911, 0.75, 'x[69] <= 0.298\ngini = 0.444\nsamples = 6\nvalue
= [2, 4]'),
 Text(0.8987341772151899, 0.7045454545454546, 'gini = 0.0\nsamples = 2\nvalue =
[2, 0]'),
 Text(0.9240506329113924, 0.7045454545454546, 'gini = 0.0\nsamples = 4\nvalue =
[0, 4]'),
 Text(0.9746835443037974, 0.7954545454545454, 'x[5] <= 0.572\ngini = 0.434\nsample
```

```
s = 22\nvalue = [7, 15]'),
 Text(0.9620253164556962, 0.75, 'x[13] <= 0.167\ngini = 0.117\nsamples = 16\nvalue
= [1, 15]'),
 Text(0.9493670886075949, 0.7045454545454546, 'gini = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.9746835443037974, 0.7045454545454546, 'gini = 0.0\nsamples = 15\nvalue =
[0, 15]'),
 Text(0.9873417721518988, 0.75, 'gini = 0.0\nsamples = 6\nvalue = [6, 0]')]
```

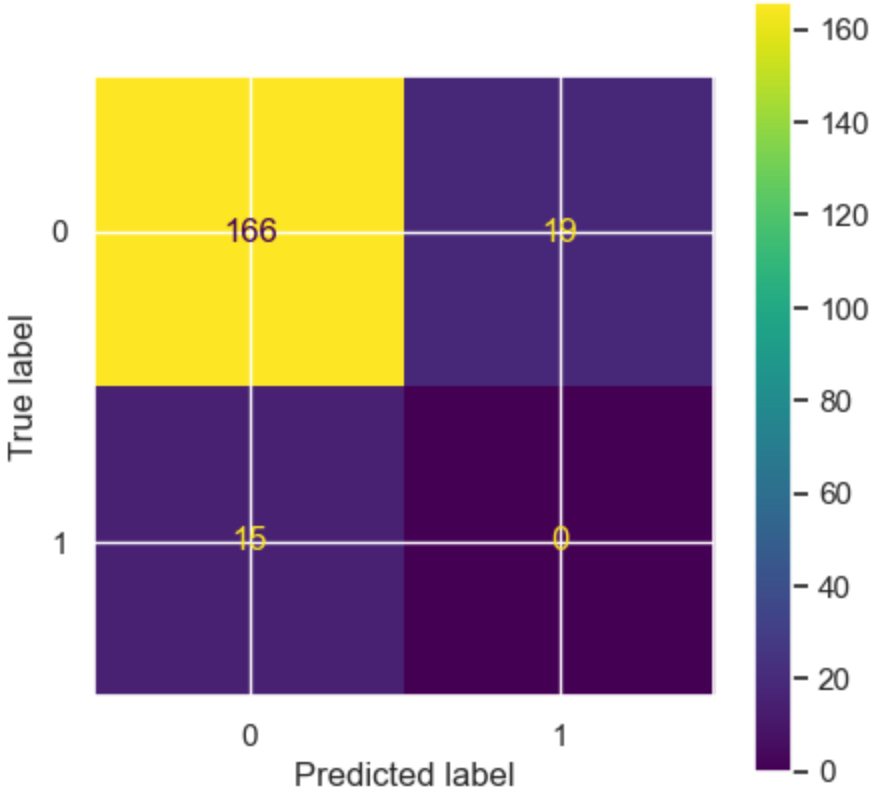


In [129…
```
y_pred_dt = dt.predict(X_test)
```

In [130…
```
print(classification_report(y_test, y_pred_dt))
```

```
              precision    recall  f1-score   support

         0.0       0.92      0.90      0.91       185
         1.0       0.00      0.00      0.00        15

    accuracy                           0.83       200
   macro avg       0.46      0.45      0.45       200
weighted avg       0.85      0.83      0.84       200
```

In [131…
```
labels = [0,1]
cm = confusion_matrix(y_test, y_pred_dt, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```



In [132…
```
#Hyperparameter Tuning
```

In [133…
```
param_grid = {
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

In [134…
```
grid = GridSearchCV(dt, param_grid, n_jobs=-1, cv=3)
grid.fit(X_train, y_train)
```

Out[134…
GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier

In [135…
```
print(grid.best_params_)
```

```
{'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5}
```

In [116…
```
dt=DecisionTreeClassifier(max_depth=None, min_samples_leaf=1, min_samples_split=5)
dt.fit(X_train, y_train)
```

Out[116…
DecisionTreeClassifier
DecisionTreeClassifier(min_samples_split=5)

In [117…
```
y_pred_dt = dt.predict(X_test)
```

In [118…
```
print(classification_report(y_test, y_pred_dt))
```

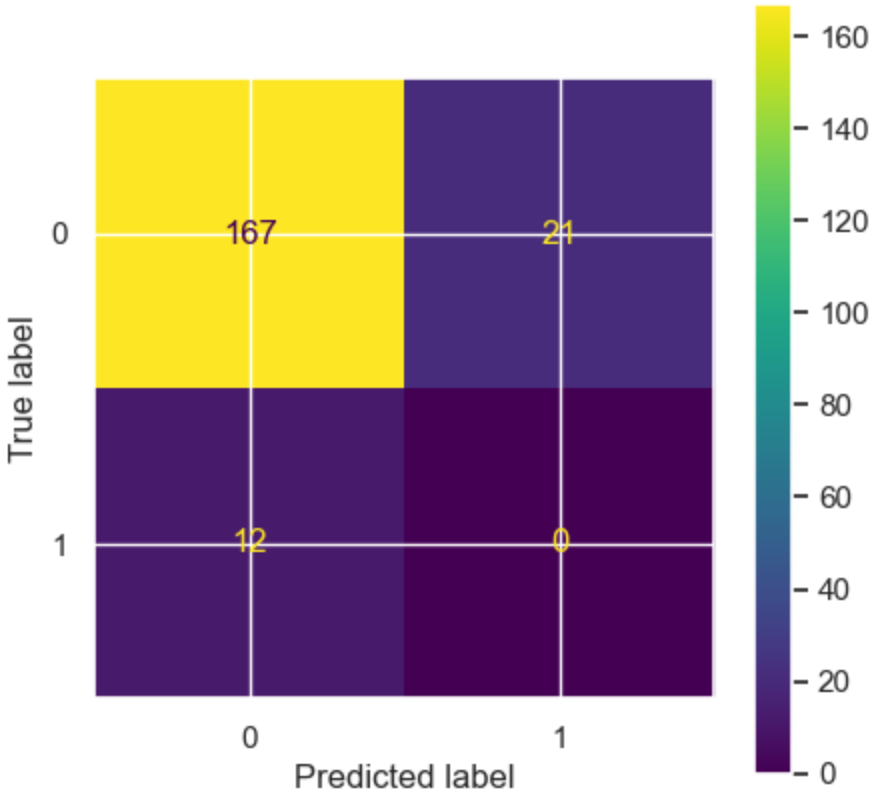
```
              precision    recall  f1-score   support

         0.0       0.93      0.89      0.91       188
         1.0       0.00      0.00      0.00        12

    accuracy                           0.83       200
   macro avg       0.47      0.44      0.46       200
weighted avg       0.88      0.83      0.86       200
```

In [119…
```
acc_dt=metrics.accuracy_score(y_test, y_pred_dt)
```

In [120…
```
auc_dt=metrics.roc_auc_score(y_test, y_pred_dt)
print(auc_dt)
```

```
0.4441489361702128
```

In [121…
```
labels = [0,1]
cm = confusion_matrix(y_test, y_pred_dt, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```



In [142…
```
# 5) LSTM
```

In [143…
```
model = Sequential()
```

In [144…
```
print(X_train.shape , y_train.shape)
```

```
(1494, 94) (1494,)
```

In [141…
```
#param_grid = {'neurons': [8, 16, 32, 64, 128],
 #           'optimizer': ['SGD', 'RMSprop', 'Adam'],
  #          'activation':['relu', 'tanh', 'sigmoid', 'linear','swish']}
```

In [142…
```
#grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=3)
#grid_search_result = grid_search.fit(X_train, y_train)
```

In [145…
```
X_train_lstm = X_train.values.reshape(1494, 94, 1)
X_train_lstm.shape
```

Out[145… (1494, 94, 1)

In [146…
```
model = Sequential()

model.add(LSTM(128, return_sequences=True, activation='relu', input_shape=(94, 1)))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [147…
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 94, 128)           66560

 dense (Dense)               (None, 94, 64)            8256

 dense_1 (Dense)             (None, 94, 32)            2080

 dense_2 (Dense)             (None, 94, 1)             33

=================================================================
Total params: 76929 (300.50 KB)
Trainable params: 76929 (300.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [148…
```
model.compile(optimizer = 'Adam', loss = 'binary_crossentropy', metrics = ['accurac
```

In [149…
```
model.fit(X_train_lstm, y_train)
```

```
47/47 [==============================] - 7s 68ms/step - loss: 0.6934 - accuracy: 0.4
927
```

Out[149… <keras.src.callbacks.History at 0x29b6ca91700>

In [150…
```
y_pred_lstm = model.predict(X_test)
```

```
7/7 [==============================] - 0s 26ms/step
```

In [151…
```
model.save('LSTM.p5')
```

```
INFO:tensorflow:Assets written to: LSTM.p5\assets
INFO:tensorflow:Assets written to: LSTM.p5\assets
```

In [152…
```
# model evaluation
from keras.models import load_model

model = load_model('LSTM.p5')
scores = model.evaluate(X_test, y_test)

acc_lstm = scores[1]

print('Test accuracy: ', scores[1])
```

```
7/7 [==============================] - 0s 26ms/step - loss: 0.6888 - accuracy: 0.907
7
Test accuracy:  0.9076595902442932
```

In [154…
```
from sklearn.metrics import confusion_matrix

#Predict
y_prediction = model.predict(X_test)

#Create confusion matrix and normalizes it over predicted (columns)
#result = confusion_matrix(y_test, y_prediction , normalize='pred')
```

```
7/7 [==============================] - 0s 26ms/step
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[154], line 7
      4 y_prediction = model.predict(X_test)
      6 #Create confusion matrix and normalizes it over predicted (columns)
----> 7 result = confusion_matrix(y_test, y_prediction , normalize='pred')

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:317, in confus
ion_matrix(y_true, y_pred, labels, sample_weight, normalize)
    232 def confusion_matrix(
    233     y_true, y_pred, *, labels=None, sample_weight=None, normalize=None
    234 ):
    235     """Compute confusion matrix to evaluate the accuracy of a classificatio
n.
    236
    237     By definition a confusion matrix :math:`C` is such that :math:`C_{i, j}`
   (...)
    315     (0, 2, 1, 1)
    316     """
--> 317     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    318     if y_type not in ("binary", "multiclass"):
    319         raise ValueError("%s is not supported" % y_type)

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:95, in _check_
targets(y_true, y_pred)
     92     y_type = {"multiclass"}
     94 if len(y_type) > 1:
---> 95     raise ValueError(
     96         "Classification metrics can't handle a mix of {0} and {1} targets".f
ormat(
     97             type_true, type_pred
     98         )
     99     )
    101 # We can't have more than one value on y_type => The set is no more needed
    102 y_type = y_type.pop()

ValueError: Classification metrics can't handle a mix of binary and unknown targets
```

In [155…

```
#auc_lstm=metrics.roc_auc_score(y_test, y_prediction)
#print(auc_lstm)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[155], line 1
----> 1 auc_lstm=metrics.roc_auc_score(y_test, y_prediction)
      2 print(auc_lstm)

File ~\anaconda3\lib\site-packages\sklearn\metrics\_ranking.py:551, in roc_auc_score
(y_true, y_score, average, sample_weight, max_fpr, multi_class, labels)
    549 y_type = type_of_target(y_true, input_name="y_true")
    550 y_true = check_array(y_true, ensure_2d=False, dtype=None)
--> 551 y_score = check_array(y_score, ensure_2d=False)
    553 if y_type == "multiclass" or (
    554     y_type == "binary" and y_score.ndim == 2 and y_score.shape[1] > 2
    555 ):
    556     # do not support partial ROC computation for multiclass
    557     if max_fpr is not None and max_fpr != 1.0:

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:915, in check_array(a
rray, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensu
re_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator, input_name)
    910     raise ValueError(
    911         "dtype='numeric' is not compatible with arrays of bytes/strings."
    912         "Convert your data to numeric values explicitly instead."
    913     )
    914 if not allow_nd and array.ndim >= 3:
--> 915     raise ValueError(
    916         "Found array with dim %d. %s expected <= 2."
    917         % (array.ndim, estimator_name)
    918     )
    920 if force_all_finite:
    921     _assert_all_finite(
    922         array,
    923         input_name=input_name,
    924         estimator_name=estimator_name,
    925         allow_nan=force_all_finite == "allow-nan",
    926     )

ValueError: Found array with dim 3. None expected <= 2.
```

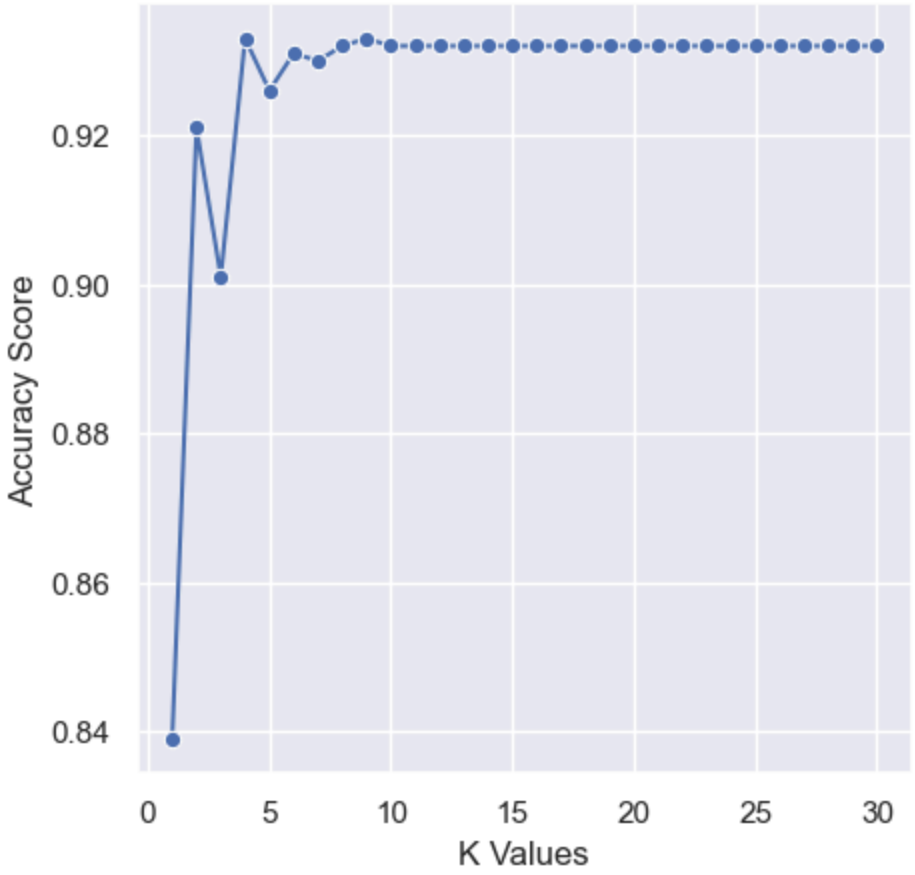
In [156…
```
# 6) KNN
```

In [122…
```
#using cross validation to find best value for k
k_values = [i for i in range (1,31)]
scores = []

scaler = StandardScaler()
X = scaler.fit_transform(X)

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    score = cross_val_score(knn, X, y, cv=5)
    scores.append(np.mean(score))
```

In [123…
```
sns.lineplot(x = k_values, y = scores, marker = 'o')
plt.xlabel("K Values")
```

```
plt.ylabel("Accuracy Score")
```

Out[123… Text(0, 0.5, 'Accuracy Score')



In [124…
```
best_index = np.argmax(scores)
best_k = k_values[best_index]
```

In [125…
```
best_k
```

Out[125… 4

In [126…
```
knn = KNeighborsClassifier(n_neighbors=4)
```

In [127…
```
knn.fit(X_train, y_train)
```

Out[127…
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=4)
```

In [128…
```
y_pred_knn = knn.predict(X_test)
```

In [129…
```
print(classification_report(y_test, y_pred_knn))
```

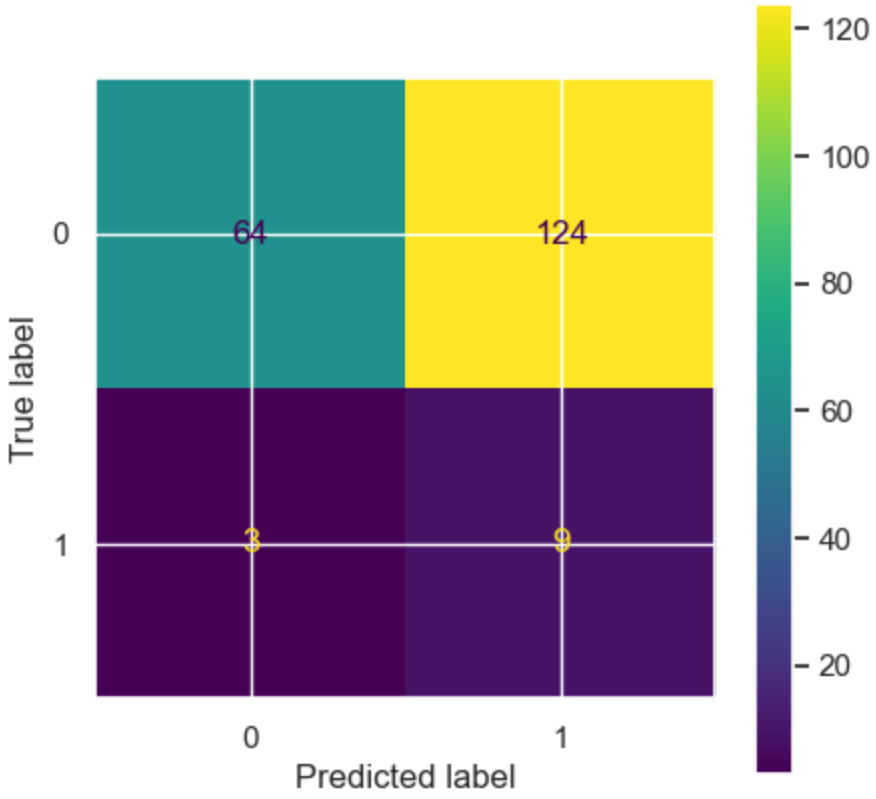
```
                precision    recall  f1-score   support

           0.0       0.96      0.34      0.50       188
           1.0       0.07      0.75      0.12        12

      accuracy                           0.36       200
     macro avg       0.51      0.55      0.31       200
  weighted avg       0.90      0.36      0.48       200
```

In [130…
```
acc_knn=metrics.accuracy_score(y_test, y_pred_knn)
```

In [131…
```
auc_knn=metrics.roc_auc_score(y_test, y_pred_knn)
print(auc_knn)
```

```
0.5452127659574468
```

In [132…
```
labels = [0,1]
cm = confusion_matrix(y_test, y_pred_knn, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```



In [168…
```
# 7) Random Forest
```

In [169…
```
param_grid = {
    'n_estimators': [25, 50, 100, 150],
    'max_features': ['sqrt', 'log2', None],
    'max_depth': [3, 6, 9],
    'max_leaf_nodes': [3, 6, 9],
}
```

In [170…
```
grid_search = GridSearchCV(RandomForestClassifier(),
                           param_grid=param_grid)
grid_search.fit(X_train, y_train)
print(grid_search.best_estimator_)
```

```
RandomForestClassifier(max_depth=9, max_features='log2', max_leaf_nodes=9,
                       n_estimators=150)
```

In [133…
```
RF = RandomForestClassifier(max_depth=9, max_features="log2", max_leaf_nodes=9, n_e
RF.fit(X_train, y_train)
y_pred_RF = RF.predict(X_test)
print(classification_report(y_pred_RF, y_test))
```

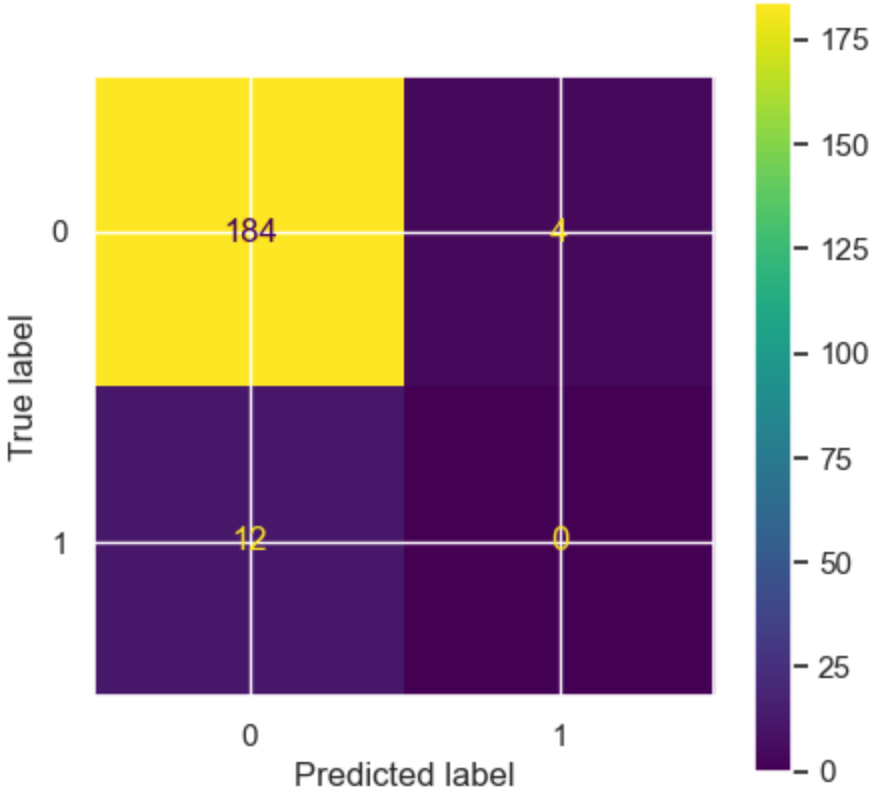
```
              precision    recall  f1-score   support

         0.0       0.98      0.94      0.96       196
         1.0       0.00      0.00      0.00         4

    accuracy                           0.92       200
   macro avg       0.49      0.47      0.48       200
weighted avg       0.96      0.92      0.94       200
```

In [134…
```
acc_RF=metrics.accuracy_score(y_test, y_pred_RF)
```

In [135…
```
auc_RF=metrics.roc_auc_score(y_test, y_pred_RF)
print(auc_RF)
```

```
0.48936170212765956
```

In [136…
```
labels = [0,1]
cm = confusion_matrix(y_test, y_pred_RF, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```



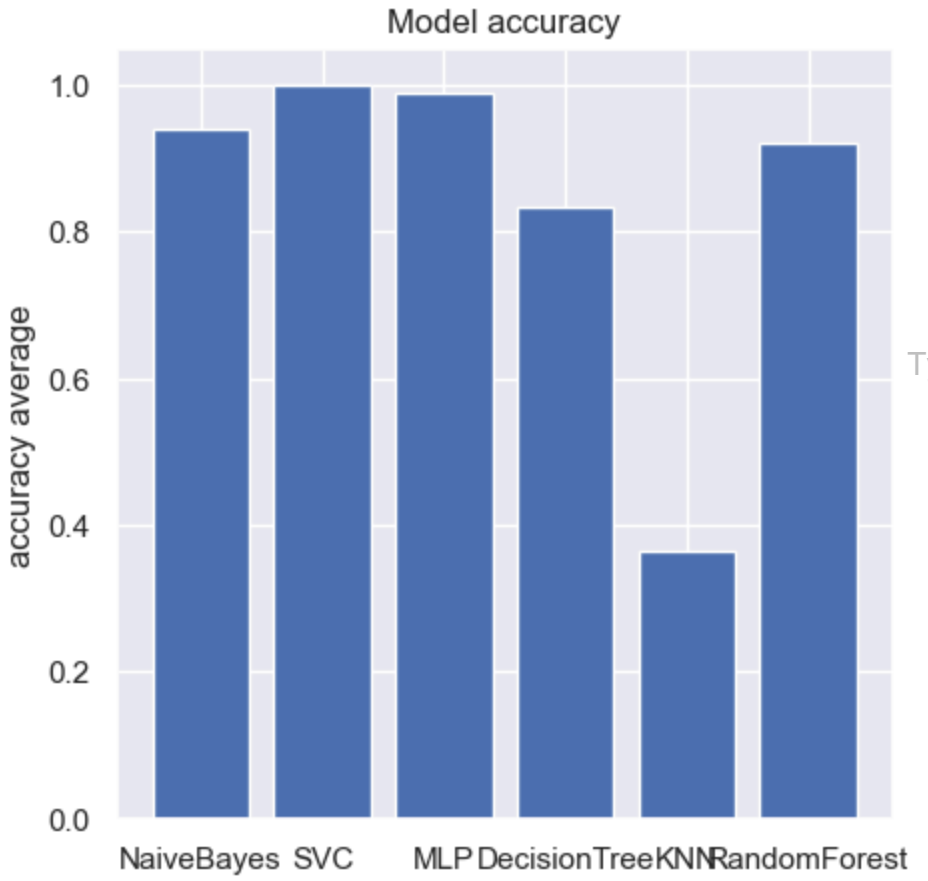
In [137…
```
#CrossVALIDATION
result = cross_val_score(RF, X_test, y_test, cv=10)
print(result)
```

```
[0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.9  0.9 ]
```

In [176…
```
#Summary of the individual models
```

In [138…
```
fig, ax = plt.subplots()

Models = ["NaiveBayes", "SVC", "MLP", "DecisionTree", "KNN", "RandomForest"]
counts = [acc_cnb, acc_svc, acc_mlp, acc_dt, acc_knn, acc_RF]

ax.bar(Models, counts)

ax.set_ylabel("accuracy average")
ax.set_title("Model accuracy")
```

Out[138…
```
Text(0.5, 1.0, 'Model accuracy')
```



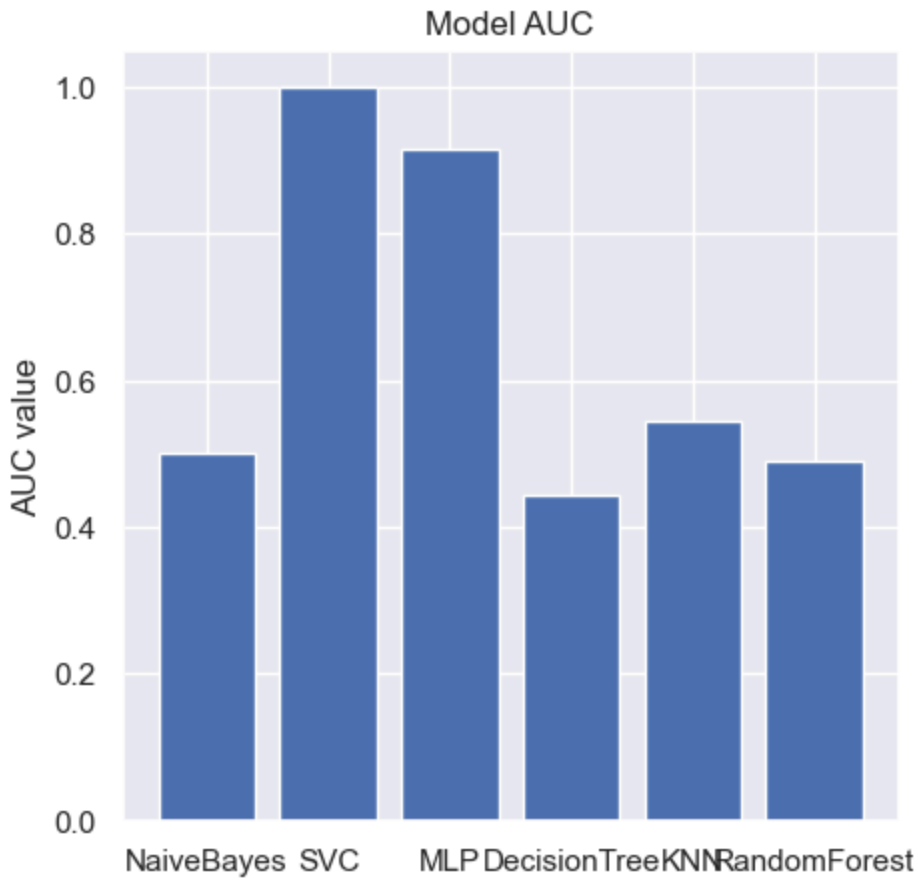
Type text here

In [141…

```
fig, ax = plt.subplots()

Models = ["NaiveBayes", "SVC", "MLP", "DecisionTree", "KNN", "RandomForest"]
counts = [auc_cnb, auc_svc, auc_mlp, auc_dt, auc_knn, auc_RF]

ax.bar(Models, counts)

ax.set_ylabel("AUC value")
ax.set_title("Model AUC")
```

Out[141…

```
Text(0.5, 1.0, 'Model AUC')
```



In [181…
```
#ENSEMBLE LEARNING
```

In [142…
```
#Simple averaging
pred_final = ((acc_cnb + acc_svc + acc_mlp + acc_dt + acc_RF) / 5)
pred_final
```

Out[142… 0.9369999999999999

In [183…
```
#BOOSTING
```

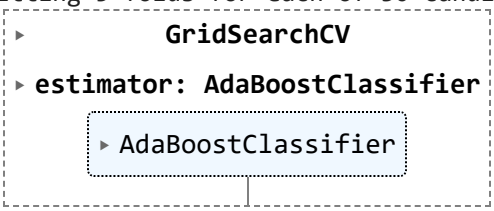
In [143…
```
# boosting
ada = AdaBoostClassifier()

# training
ada.fit(X_train, y_train)

# prediction
prediction = ada.predict(X_test)

# evaluation
accuracy = round(accuracy_score(y_test, prediction) * 100, 3)
auc = round(roc_auc_score(y_test, prediction), 3)

print(f" Accuracy: {accuracy}%")
print(f" AUC score: {auc}")
```

```
Accuracy: 100.0%
AUC score: 1.0
```

In [185…
```
#Hyperparameter tuning
param_grid = {
    'n_estimators' : [50, 70, 90, 120, 180, 200],
    'learning_rate' : [0.001, 0.01, 0.1, 1, 10]
}

grid_search = GridSearchCV(ada, param_grid, n_jobs = -1, cv = 5, verbose = 1)
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
```

Out[185…



In [186…
```
print(grid_search.best_params_)
```

```
{'learning_rate': 0.1, 'n_estimators': 90}
```

In [144…
```
ada = AdaBoostClassifier(learning_rate=0.00001, n_estimators=90)
ada.fit(X_train, y_train)
prediction = ada.predict(X_test)
accuracy = round(accuracy_score(y_test, prediction) * 100, 3)
auc = round(roc_auc_score(y_test, prediction), 3)
print(f" Accuracy: {accuracy}%")
print(f" AUC score: {auc}")
```

```
Accuracy: 49.5%
AUC score: 0.497
```

In [ ]:
```

```

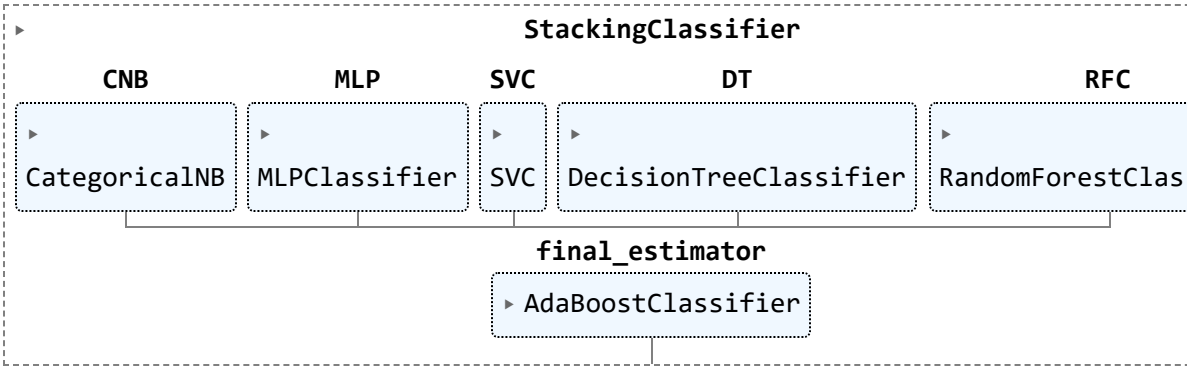
In [145…
```
#Stacking and boosting
estimators = []
estimators.append(('CNB', CategoricalNB()))
estimators.append(("MLP", MLPClassifier(max_iter=500, activation="tanh", alpha=0.00
estimators.append(("SVC", SVC(C=0.1, gamma=1, kernel="linear")))
estimators.append(("DT", DecisionTreeClassifier(max_depth=None, min_samples_leaf=1,
#estimators.append(("KNN", KNeighborsClassifier(n_neighbors=8)))
estimators.append(("RFC", RandomForestClassifier(max_depth=9, max_features="log2",

# building the level 1 model
ADA = AdaBoostClassifier(learning_rate=0.00001, n_estimators=90)
```

In [ ]:
```

```

In [146…
```
SC = StackingClassifier(estimators=estimators,final_estimator=ADA)
```

In [147…
```
SC.fit(X_train, y_train)
```

Out[147…



In [148…
```
prediction = SC.predict(X_test)
```

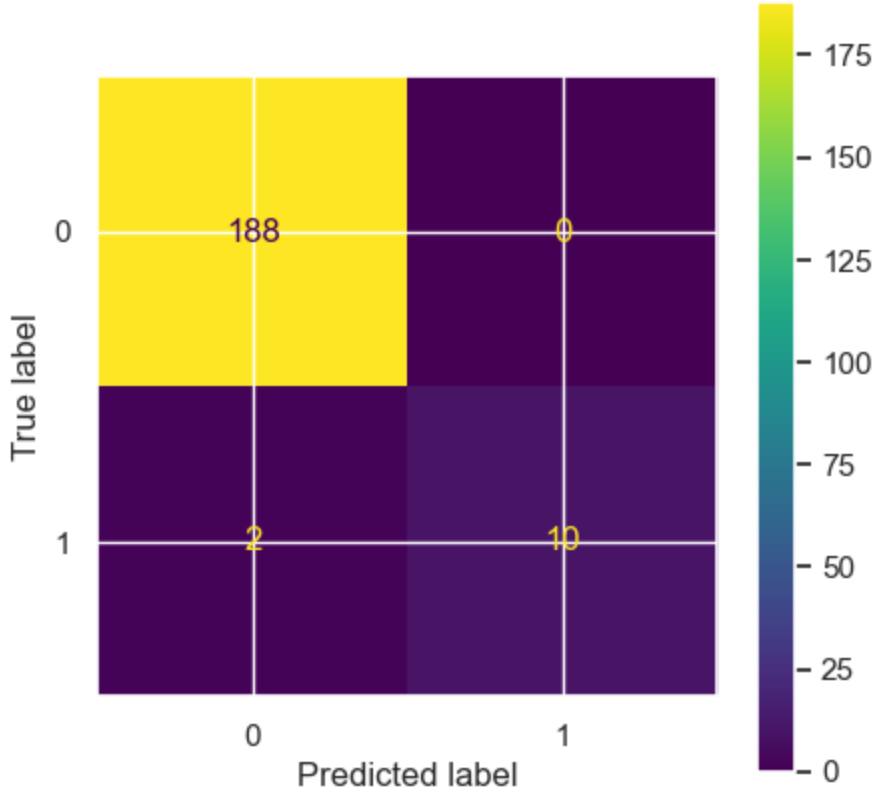
In [149…
```
print(classification_report(y_test, prediction))
```

```
              precision    recall  f1-score   support

         0.0       0.99      1.00      0.99       188
         1.0       1.00      0.83      0.91        12

    accuracy                           0.99       200
   macro avg       0.99      0.92      0.95       200
weighted avg       0.99      0.99      0.99       200
```

In [150…
```
accuracy = round(accuracy_score(y_test, prediction) * 100, 3)
auc = round(roc_auc_score(y_test, prediction), 3)

print(f" Accuracy: {accuracy}%")
print(f" AUC score: {auc}")
```

```
Accuracy: 99.0%
AUC score: 0.917
```

In [151…
```
labels = [0,1]
cm = confusion_matrix(y_test, prediction, labels=labels)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels)
disp.plot();
```


188
0
2
10
0
1
True label
Predicted label
175
150
125
100
75
50
25
0

In [ ]:

In [ ]: