FraudDetection_with_Preprocessing

November 3, 2024

Name: Hamid El Messaoudi

Date: 11/3/2024

Program: Complete Data Preprocessing Workflow for Fraud Detection

[55]:
```python
# Importing the basic libraries we need for data manipulation and analysis
import pandas as pd
import numpy as np
```

[56]:
```python
# Installing necessary packages (only run if not installed)
!python -m pip install --upgrade pip
!pip install sklearn_pandas xgboost
```

```
Requirement already satisfied: pip in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages
(24.3.1)
Requirement already satisfied: sklearn_pandas in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (2.2.0)
Requirement already satisfied: xgboost in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (2.1.2)
Requirement already satisfied: scikit-learn>=0.23.0 in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (from
sklearn_pandas) (1.4.0)
Requirement already satisfied: scipy>=1.5.1 in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (from
sklearn_pandas) (1.12.0)
Requirement already satisfied: pandas>=1.1.4 in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (from
sklearn_pandas) (2.2.0)
Requirement already satisfied: numpy>=1.18.1 in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (from
sklearn_pandas) (1.26.3)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (from
pandas>=1.1.4->sklearn_pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (from
pandas>=1.1.4->sklearn_pandas) (2024.1)
```

```
Requirement already satisfied: tzdata>=2022.7 in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (from
pandas>=1.1.4->sklearn_pandas) (2023.4)
Requirement already satisfied: joblib>=1.2.0 in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (from
scikit-learn>=0.23.0->sklearn_pandas) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (from
scikit-learn>=0.23.0->sklearn_pandas) (3.2.0)
Requirement already satisfied: six>=1.5 in
c:\users\hamid\appdata\local\programs\python\python312\lib\site-packages (from
python-dateutil>=2.8.2->pandas>=1.1.4->sklearn_pandas) (1.16.0)
```

[57]:
```
# Reading the dataset
data = pd.read_csv('insuranceFraud.csv')
```

[58]:
```
# Having a look at the data
data.head()
```

[58]:
```
   months_as_customer  age  policy_number policy_bind_date policy_state  \
0                 328   48         521585       10/17/2014           OH
1                 228   42         342868        6/27/2006           IN
2                 134   29         687698         9/6/2000           OH
3                 256   41         227811        5/25/1990           IL
4                 228   44         367455         6/6/2014           IL

  policy_csl  policy_deductable  policy_annual_premium  umbrella_limit  \
0    250/500               1000                1406.91               0
1    250/500               2000                1197.22         5000000
2    100/300               2000                1413.14         5000000
3    250/500               2000                1415.74         6000000
4   500/1000               1000                1583.91         6000000

   insured_zip  … witnesses police_report_available total_claim_amount  \
0       466132  …         2                     YES              71610
1       468176  …         0                       ?               5070
2       430632  …         3                      NO              34650
3       608117  …         2                      NO              63400
4       610706  …         1                      NO               6500

  injury_claim property_claim  vehicle_claim  auto_make auto_model auto_year  \
0         6510          13020          52080       Saab        92x      2004
1          780            780           3510   Mercedes       E400      2007
2         7700           3850          23100      Dodge        RAM      2007
3         6340           6340          50720  Chevrolet      Tahoe      2014
4         1300            650           4550     Accura        RSX      2009
```

```
  fraud_reported
0              Y
1              Y
2              N
3              Y
4              N

[5 rows x 39 columns]
```

[59]:
```python
# Replacing all the "?" values with NaN to make them easier to handle
data = data.replace('?', np.nan)
```

[60]:
```python
# list of columns not necessary for pfrediction
cols_to_drop=['policy_number','policy_bind_date','policy_state','insured_zip','incident_locati
```

[61]:
```python
# dropping the unnecessary columns
data.drop(columns=cols_to_drop,inplace=True)
```

[62]:
```python
# checking the data after dropping the columns
data.head()
```

[62]:
```
   months_as_customer  age policy_csl  policy_deductable  \
0                 328   48    250/500               1000
1                 228   42    250/500               2000
2                 134   29    100/300               2000
3                 256   41    250/500               2000
4                 228   44   500/1000               1000

   policy_annual_premium  umbrella_limit insured_sex insured_education_level  \
0                1406.91               0        MALE                      MD
1                1197.22         5000000        MALE                      MD
2                1413.14         5000000      FEMALE                     PhD
3                1415.74         6000000      FEMALE                     PhD
4                1583.91         6000000        MALE               Associate

  insured_occupation insured_relationship  …  number_of_vehicles_involved  \
0       craft-repair              husband  …                            1
1  machine-op-inspct       other-relative  …                            1
2              sales            own-child  …                            3
3       armed-forces            unmarried  …                            1
4              sales            unmarried  …                            1

   property_damage bodily_injuries witnesses police_report_available  \
0              YES               1         2                      YES
1              NaN               0         0                      NaN
2               NO               2         3                       NO
3              NaN               1         2                       NO
```

```
4            NO           0        1                      NO

  total_claim_amount  injury_claim  property_claim vehicle_claim  \
0              71610          6510           13020         52080
1               5070           780             780          3510
2              34650          7700            3850         23100
3              63400          6340            6340         50720
4               6500          1300             650          4550

   fraud_reported
0               Y
1               Y
2               N
3               Y
4               N

[5 rows x 27 columns]
```

[63]:

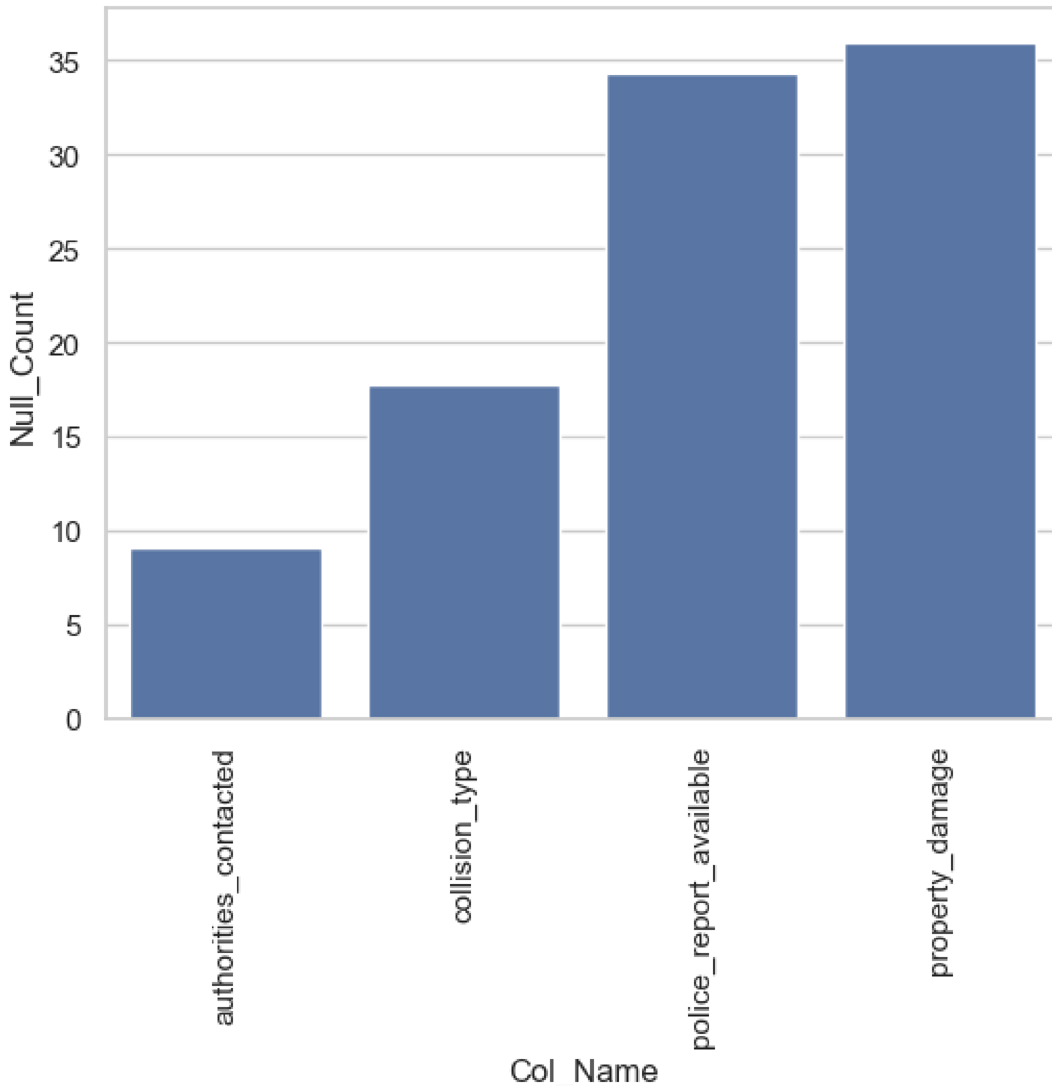
```python
# checking for missing values
data.isna().sum()
```

[63]:

```
months_as_customer               0
age                              0
policy_csl                       0
policy_deductable                0
policy_annual_premium            0
umbrella_limit                   0
insured_sex                      0
insured_education_level          0
insured_occupation               0
insured_relationship             0
capital-gains                    0
capital-loss                     0
incident_type                    0
collision_type                 178
incident_severity                0
authorities_contacted           91
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
```

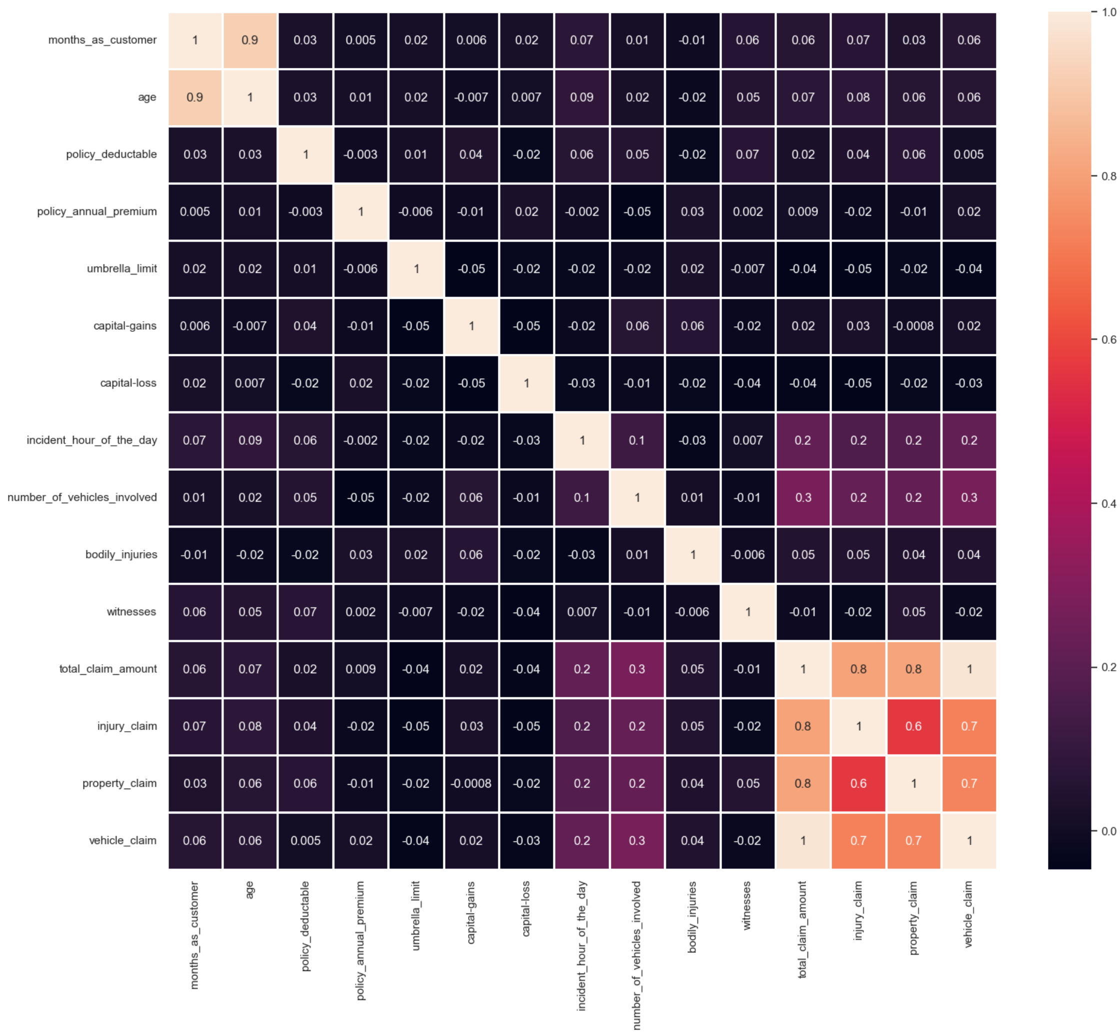
```
fraud_reported                    0
dtype: int64
```

[64]:
```
import seaborn as sns
import matplotlib.pyplot as plt
missing = data.isnull().mean() * 100  # percentage
missing = missing[missing > 0]
missing.sort_values(inplace=True)
missing = missing.to_frame()
missing.columns = ['Null_Count']
missing.index.names = ['Col_Name']
missing = missing.reset_index()

sns.set(style='whitegrid', color_codes=True)
sns.barplot(x='Col_Name', y='Null_Count', data=missing)
plt.xticks(rotation=90)
plt.show()
```



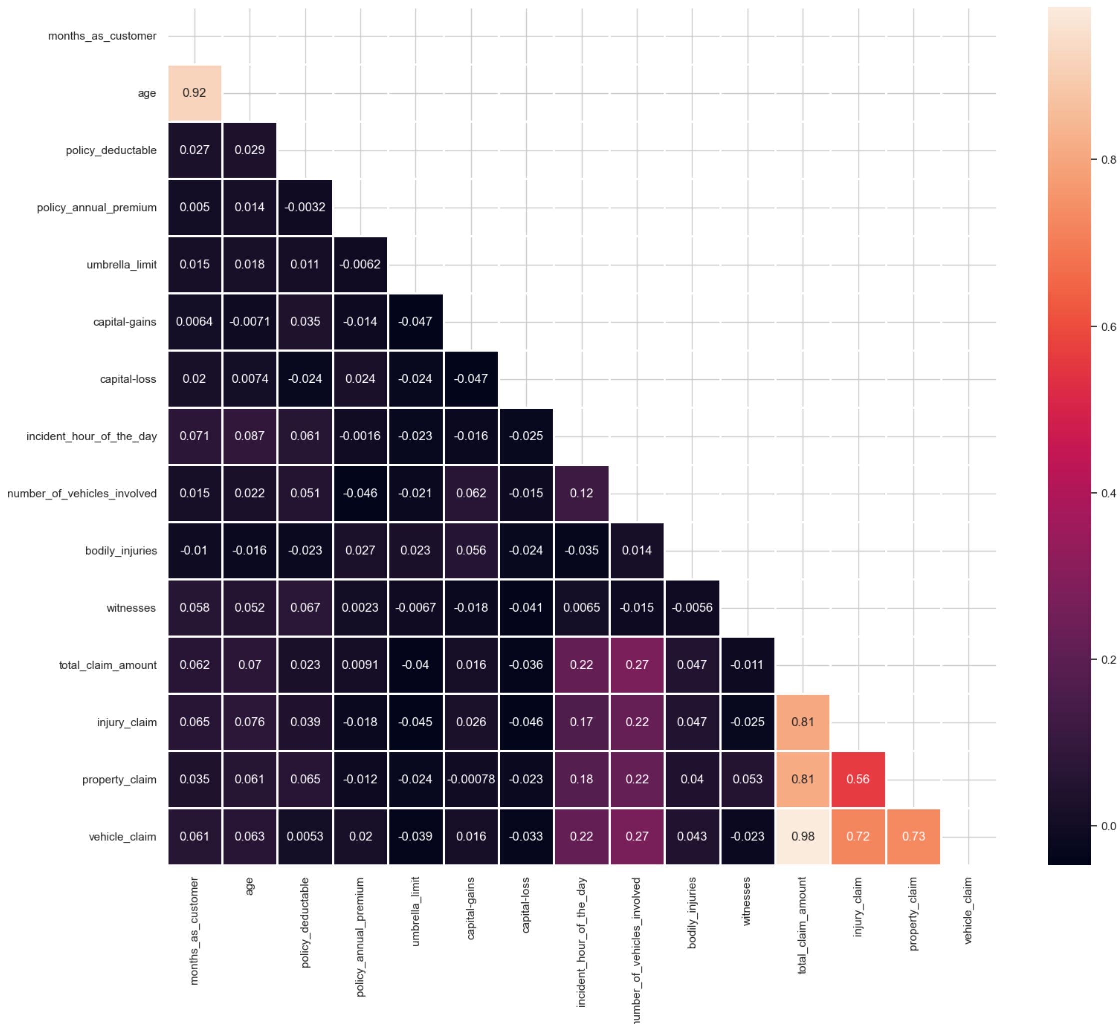
[65]:

```
# Set the figure size for the correlation heatmap
plt.figure(figsize=(18, 15))

# Calculate the correlation matrix for numerical columns in the data
corr = data.select_dtypes(include=['number']).corr()

# Plot the heatmap of the correlation matrix with annotations and specified␣
↪formatting
sns.heatmap(data=corr, annot=True, fmt='.1g', linewidth=2)

# Display the plot
plt.show()
```



```
[66]: # Checking if any of our numerical features are strongly related to each other␣
       ↪(aka, correlated).
      # If two features are super similar, we might not need both.
      # This just gives us a sense of what's related in our data.
      plt.figure(figsize = (18,15))
      corr = data.select_dtypes(include=['number']).corr()
      mask = np.triu(np.ones_like(corr,dtype =bool))

      sns.heatmap(data =corr, mask = mask, annot = True, fmt ='.2g', linewidth =1)
      plt.show
```

```
[66]: <function matplotlib.pyplot.show(close=None, block=None)>
```



[67]:
```python
data.drop(columns = ['age', 'total_claim_amount'], inplace = True, axis = 1)
data.head()
```

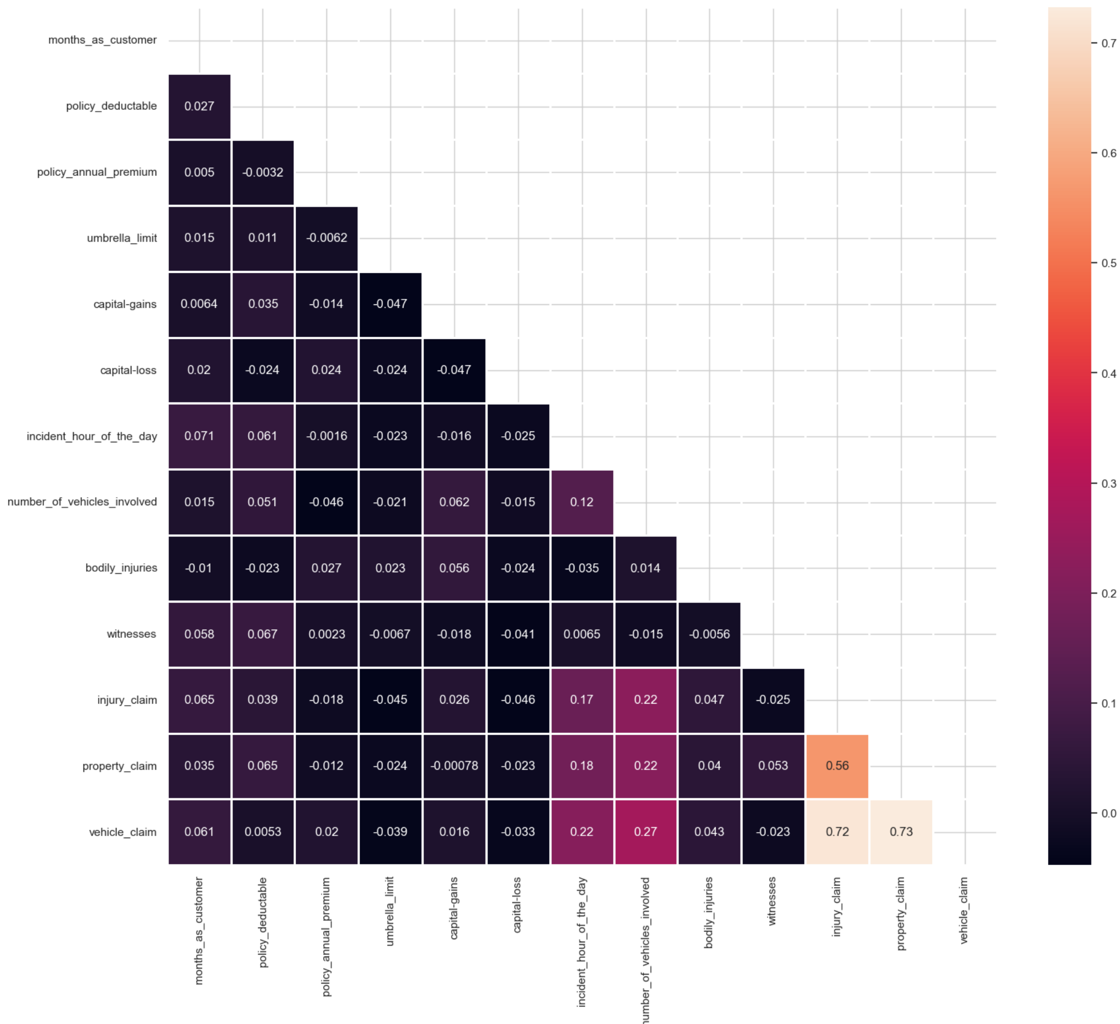
[67]:
```
   months_as_customer policy_csl  policy_deductable  policy_annual_premium  \
0                 328    250/500               1000                1406.91
1                 228    250/500               2000                1197.22
2                 134    100/300               2000                1413.14
3                 256    250/500               2000                1415.74
4                 228   500/1000               1000                1583.91

   umbrella_limit insured_sex insured_education_level insured_occupation  \
0               0        MALE                      MD       craft-repair
1         5000000        MALE                      MD  machine-op-inspct
2         5000000      FEMALE                     PhD              sales
3         6000000      FEMALE                     PhD       armed-forces
```

```
4         6000000        MALE              Associate               sales

  insured_relationship  capital-gains  …  incident_hour_of_the_day  \
0              husband          53300  …                         5
1       other-relative              0  …                         8
2            own-child          35100  …                         7
3            unmarried          48900  …                         5
4            unmarried          66000  …                        20

  number_of_vehicles_involved property_damage bodily_injuries witnesses  \
0                           1             YES               1         2
1                           1             NaN               0         0
2                           3              NO               2         3
3                           1             NaN               1         2
4                           1              NO               0         1

   police_report_available  injury_claim property_claim  vehicle_claim  \
0                      YES          6510          13020          52080
1                      NaN           780            780           3510
2                       NO          7700           3850          23100
3                       NO          6340           6340          50720
4                       NO          1300            650           4550

   fraud_reported
0               Y
1               Y
2               N
3               Y
4               N

[5 rows x 25 columns]
```

[68]:

```python
plt.figure(figsize = (18,15))
corr = data.select_dtypes(include=['number']).corr()
mask = np.triu(np.ones_like(corr,dtype =bool))

sns.heatmap(data =corr, mask = mask, annot = True, fmt ='.2g', linewidth =1)
plt.show
```

[68]: <function matplotlib.pyplot.show(close=None, block=None)>



[69]:

```
# Importing SimpleImputer from sklearn
from sklearn.impute import SimpleImputer

# Creating an imputer instance to fill missing values with the most frequent␣
↪value in each column
imputer = SimpleImputer(strategy='most_frequent')

# Imputing missing values for specific categorical columns
data['collision_type'] = imputer.fit_transform(data[['collision_type']]).ravel()
data['property_damage'] = imputer.fit_transform(data[['property_damage']]).
↪ravel()
data['police_report_available'] = imputer.
↪fit_transform(data[['police_report_available']]).ravel()
```

[70]:
```python
# As the columns which have missing values, they are only categorical, we'll␣
↪use the categorical imputer
# Importing the categorical imputer
from sklearn.impute import SimpleImputer

# Create an imputer instance for categorical columns
imputer = SimpleImputer(strategy='most_frequent')
```

[71]:
```python
# Imputing missing values for each categorical column
data['collision_type'] = imputer.fit_transform(data[['collision_type']]).ravel()
data['property_damage'] = imputer.fit_transform(data[['property_damage']]).
↪ravel()
data['police_report_available'] = imputer.
↪fit_transform(data[['police_report_available']]).ravel()
```

[72]:
```python
# Extracting the categorical columns
cat_df = data.select_dtypes(include=['object']).copy()
```

[73]:
```python
cat_df.columns
```

[73]:
```
Index(['policy_csl', 'insured_sex', 'insured_education_level',
       'insured_occupation', 'insured_relationship', 'incident_type',
       'collision_type', 'incident_severity', 'authorities_contacted',
       'property_damage', 'police_report_available', 'fraud_reported'],
      dtype='object')
```

[74]:
```python
cat_df.head()
```

[74]:
```
  policy_csl insured_sex insured_education_level insured_occupation  \
0    250/500        MALE                      MD       craft-repair
1    250/500        MALE                      MD  machine-op-inspct
2    100/300      FEMALE                     PhD              sales
3    250/500      FEMALE                     PhD       armed-forces
4   500/1000        MALE               Associate              sales

  insured_relationship            incident_type   collision_type  \
0              husband  Single Vehicle Collision   Side Collision
1       other-relative             Vehicle Theft   Rear Collision
2            own-child    Multi-vehicle Collision   Rear Collision
3            unmarried  Single Vehicle Collision  Front Collision
4            unmarried             Vehicle Theft   Rear Collision

  incident_severity authorities_contacted property_damage  \
0      Major Damage                Police             YES
1      Minor Damage                Police              NO
2      Minor Damage                Police              NO
3      Major Damage                Police              NO
```

```
4      Minor Damage                         NaN               NO

  police_report_available fraud_reported
0                     YES              Y
1                      NO              Y
2                      NO              N
3                      NO              Y
4                      NO              N
```

[75]:
```
cat_df.columns
```

[75]:
```
Index(['policy_csl', 'insured_sex', 'insured_education_level',
       'insured_occupation', 'insured_relationship', 'incident_type',
       'collision_type', 'incident_severity', 'authorities_contacted',
       'property_damage', 'police_report_available', 'fraud_reported'],
      dtype='object')
```

[76]:
```
cat_df['policy_csl'].unique()
```

[76]:
```
array(['250/500', '100/300', '500/1000'], dtype=object)
```

[77]:
```
cat_df['insured_sex'].unique()
```

[77]:
```
array(['MALE', 'FEMALE'], dtype=object)
```

[78]:
```
cat_df['insured_education_level'].unique()
```

[78]:
```
array(['MD', 'PhD', 'Associate', 'Masters', 'High School', 'College',
       'JD'], dtype=object)
```

[79]:
```
cat_df['insured_relationship'].unique()
```

[79]:
```
array(['husband', 'other-relative', 'own-child', 'unmarried', 'wife',
       'not-in-family'], dtype=object)
```

[80]:
```
cat_df['incident_type'].unique()
```

[80]:
```
array(['Single Vehicle Collision', 'Vehicle Theft',
       'Multi-vehicle Collision', 'Parked Car'], dtype=object)
```

[81]:
```
cat_df['collision_type'].unique()
```

[81]:
```
array(['Side Collision', 'Rear Collision', 'Front Collision'],
      dtype=object)
```

[82]:
```
cat_df['incident_severity'].unique()
```

[82]: array(['Major Damage', 'Minor Damage', 'Total Loss', 'Trivial Damage'],
      dtype=object)

[83]:
```
cat_df['authorities_contacted'].unique()
```

[83]: array(['Police', nan, 'Fire', 'Other', 'Ambulance'], dtype=object)

[84]:
```
cat_df['property_damage'].unique()
```

[84]: array(['YES', 'NO'], dtype=object)

[85]:
```
cat_df['police_report_available'].unique()
```

[85]: array(['YES', 'NO'], dtype=object)

[86]:
```
cat_df['fraud_reported'].unique()
```

[86]: array(['Y', 'N'], dtype=object)

[87]:
```
# custom mapping for encoding
cat_df['policy_csl'] = cat_df['policy_csl'].map({'100/300' : 1, '250/500' : 2.5␣
 ↪,'500/1000':5})
cat_df['insured_education_level'] = cat_df['insured_education_level'].map({'JD'␣
 ↪: 1, 'High School' : 2,'College':3,'Masters':4,'Associate':5,'MD':6,'PhD':7})
cat_df['incident_severity'] = cat_df['incident_severity'].map({'Trivial Damage'␣
 ↪: 1, 'Minor Damage' : 2,'Major Damage':3,'Total Loss':4})
cat_df['insured_sex'] = cat_df['insured_sex'].map({'FEMALE' : 0, 'MALE' : 1})
cat_df['property_damage'] = cat_df['property_damage'].map({'NO' : 0, 'YES' : 1})
cat_df['police_report_available'] = cat_df['police_report_available'].map({'NO'␣
 ↪: 0, 'YES' : 1})
cat_df['fraud_reported'] = cat_df['fraud_reported'].map({'N' : 0, 'Y' : 1})
```

[88]:
```
# auto encoding of categorical variables
for col in cat_df.
 ↪drop(columns=['policy_csl','insured_education_level','incident_severity','insured_sex','pro
 ↪columns:
   cat_df= pd.get_dummies(cat_df, columns=[col], prefix = [col],␣
 ↪drop_first=True)
```

[89]:
```
# Converting any True/False values to 1/0 to make all data numeric
cat_df = cat_df.applymap(lambda x: int(x) if isinstance(x, bool) else x)
```

```
C:\Users\hamid\AppData\Local\Temp\ipykernel_29924\720693957.py:2: FutureWarning:
DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  cat_df = cat_df.applymap(lambda x: int(x) if isinstance(x, bool) else x)
```

[90]:
```
# data after encoding
cat_df.head()
```

```
[90]:    policy_csl  insured_sex  insured_education_level  incident_severity  \
      0         2.5            1                        6                  3
      1         2.5            1                        6                  2
      2         1.0            0                        7                  2
      3         2.5            0                        7                  3
      4         5.0            1                        5                  2

         property_damage  police_report_available  fraud_reported  \
      0                1                        1               1
      1                0                        0               1
      2                0                        0               0
      3                0                        0               1
      4                0                        0               0

         insured_occupation_armed-forces  insured_occupation_craft-repair  \
      0                                0                                1
      1                                0                                0
      2                                0                                0
      3                                1                                0
      4                                0                                0

         insured_occupation_exec-managerial  …  insured_relationship_unmarried  \
      0                                   0  …                               0
      1                                   0  …                               0
      2                                   0  …                               0
      3                                   0  …                               1
      4                                   0  …                               1

         insured_relationship_wife  incident_type_Parked Car  \
      0                          0                         0
      1                          0                         0
      2                          0                         0
      3                          0                         0
      4                          0                         0

         incident_type_Single Vehicle Collision  incident_type_Vehicle Theft  \
      0                                       1                            0
      1                                       0                            1
      2                                       0                            0
      3                                       1                            0
      4                                       0                            1

         collision_type_Rear Collision  collision_type_Side Collision  \
      0                              0                              1
      1                              1                              0
      2                              1                              0
      3                              0                              0
```

```
4                               1                                 0

   authorities_contacted_Fire  authorities_contacted_Other  \
0                           0                            0
1                           0                            0
2                           0                            0
3                           0                            0
4                           0                            0

   authorities_contacted_Police
0                             1
1                             1
2                             1
3                             1
4                             0

[5 rows x 33 columns]
```

[91]:

```python
# data after encoding
cat_df.head()
```

[91]:

```
   policy_csl  insured_sex  insured_education_level  incident_severity  \
0         2.5            1                        6                  3
1         2.5            1                        6                  2
2         1.0            0                        7                  2
3         2.5            0                        7                  3
4         5.0            1                        5                  2

   property_damage  police_report_available  fraud_reported  \
0                1                        1               1
1                0                        0               1
2                0                        0               0
3                0                        0               1
4                0                        0               0

   insured_occupation_armed-forces  insured_occupation_craft-repair  \
0                                0                                1
1                                0                                0
2                                0                                0
3                                1                                0
4                                0                                0

   insured_occupation_exec-managerial  …  insured_relationship_unmarried  \
0                                   0  …                               0
1                                   0  …                               0
2                                   0  …                               0
3                                   0  …                               1
```

```
4                                     0  ...                               1

   insured_relationship_wife  incident_type_Parked Car  \
0                          0                         0
1                          0                         0
2                          0                         0
3                          0                         0
4                          0                         0

   incident_type_Single Vehicle Collision  incident_type_Vehicle Theft  \
0                                       1                            0
1                                       0                            1
2                                       0                            0
3                                       1                            0
4                                       0                            1

   collision_type_Rear Collision  collision_type_Side Collision  \
0                              0                              1
1                              1                              0
2                              1                              0
3                              0                              0
4                              1                              0

   authorities_contacted_Fire  authorities_contacted_Other  \
0                           0                            0
1                           0                            0
2                           0                            0
3                           0                            0
4                           0                            0

   authorities_contacted_Police
0                             1
1                             1
2                             1
3                             1
4                             0

[5 rows x 33 columns]
```

[92]:
```
# extracting the numerical columns
num_df = data.select_dtypes(include=['int64']).copy()
```

[93]:
```
num_df.columns
```

[93]:
```
Index(['months_as_customer', 'policy_deductable', 'umbrella_limit',
       'capital-gains', 'capital-loss', 'incident_hour_of_the_day',
       'number_of_vehicles_involved', 'bodily_injuries', 'witnesses',
```

```
       'injury_claim', 'property_claim', 'vehicle_claim'],
      dtype='object')
```

[94]:
```python
num_df.head()
```

[94]:
```
   months_as_customer  policy_deductable  umbrella_limit  capital-gains  \
0                 328               1000               0          53300
1                 228               2000         5000000              0
2                 134               2000         5000000          35100
3                 256               2000         6000000          48900
4                 228               1000         6000000          66000

   capital-loss  incident_hour_of_the_day  number_of_vehicles_involved  \
0             0                         5                            1
1             0                         8                            1
2             0                         7                            3
3        -62400                         5                            1
4        -46000                        20                            1

   bodily_injuries  witnesses  injury_claim  property_claim  vehicle_claim
0                1          2          6510           13020          52080
1                0          0           780             780           3510
2                2          3          7700            3850          23100
3                1          2          6340            6340          50720
4                0          1          1300             650           4550
```

[95]:
```python
# combining the Numerical and categorical dataframes to get the final dataset
final_df=pd.concat([num_df,cat_df], axis=1)
```

[96]:
```python
final_df.head()
```

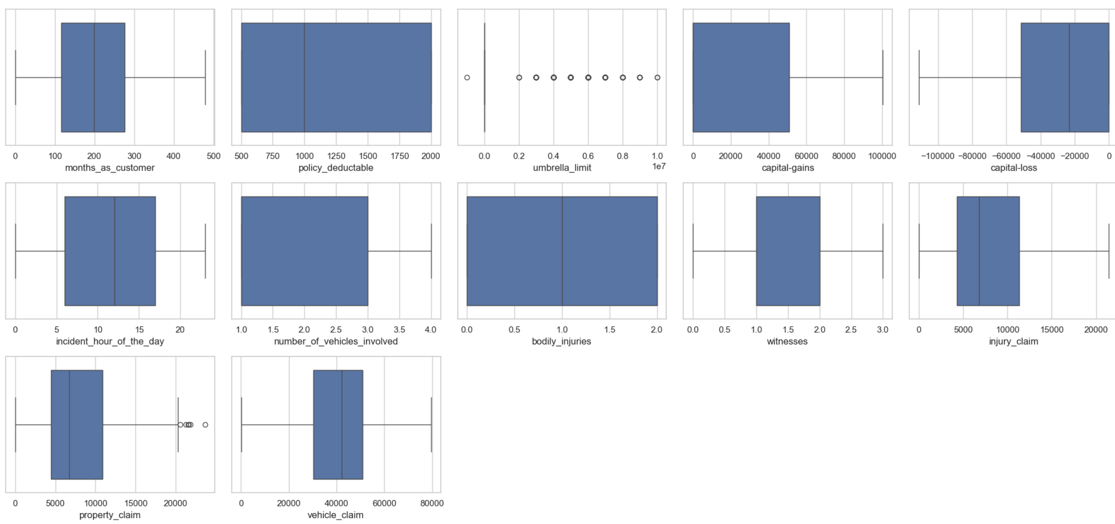
[96]:
```
   months_as_customer  policy_deductable  umbrella_limit  capital-gains  \
0                 328               1000               0          53300
1                 228               2000         5000000              0
2                 134               2000         5000000          35100
3                 256               2000         6000000          48900
4                 228               1000         6000000          66000

   capital-loss  incident_hour_of_the_day  number_of_vehicles_involved  \
0             0                         5                            1
1             0                         8                            1
2             0                         7                            3
3        -62400                         5                            1
4        -46000                        20                            1

   bodily_injuries  witnesses  injury_claim  …  \
0                1          2          6510  …
```

```
1                0          0           780  ...
2                2          3          7700  ...
3                1          2          6340  ...
4                0          1          1300  ...

   insured_relationship_unmarried  insured_relationship_wife  \
0                               0                          0
1                               0                          0
2                               0                          0
3                               1                          0
4                               1                          0

   incident_type_Parked Car  incident_type_Single Vehicle Collision  \
0                         0                                       1
1                         0                                       0
2                         0                                       0
3                         0                                       1
4                         0                                       0

   incident_type_Vehicle Theft  collision_type_Rear Collision  \
0                            0                              0
1                            1                              1
2                            0                              1
3                            0                              0
4                            1                              1

   collision_type_Side Collision  authorities_contacted_Fire  \
0                              1                           0
1                              0                           0
2                              0                           0
3                              0                           0
4                              0                           0

   authorities_contacted_Other  authorities_contacted_Police
0                            0                             1
1                            0                             1
2                            0                             1
3                            0                             1
4                            0                             0

[5 rows x 45 columns]
```

[ ]:

[97]:
```
# Checking for outliers in numerical columns

import matplotlib.pyplot as plt
```

```python
import seaborn as sns

plt.figure(figsize=(20, 15))
plot_number = 1

for col in num_df.columns:
    if plot_number <= 24:  # Limiting to 24 plots for readability
        ax = plt.subplot(5, 5, plot_number)  # Creating a 5x5 grid of boxplots
        sns.boxplot(x=num_df[col], ax=ax)
        plt.xlabel(col, fontsize=12)
        plot_number += 1

plt.tight_layout()
plt.show()
```



[98]:
```python
# Checking for potential outliers using IQR, without removing them yet
outliers_summary = {}

for col in num_df:
    Q1 = final_df[col].quantile(0.25)
    Q3 = final_df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers_count = final_df[(final_df[col] < lower_bound) | (final_df[col] > upper_bound)].shape[0]

    # Storing summary information
    outliers_summary[col] = {
```

```
        "Lower Bound": lower_bound,
        "Upper Bound": upper_bound,
        "Outliers Count": outliers_count
    }

# Displaying the outliers summary
pd.DataFrame(outliers_summary).T
```

[98]:

	Lower Bound	Upper Bound	Outliers Count
months_as_customer	-125.0	517.0	0.0
policy_deductable	-1750.0	4250.0	0.0
umbrella_limit	0.0	0.0	202.0
capital-gains	-76537.5	127562.5	0.0
capital-loss	-128750.0	77250.0	0.0
incident_hour_of_the_day	-10.5	33.5	0.0
number_of_vehicles_involved	-2.0	6.0	0.0
bodily_injuries	-3.0	5.0	0.0
witnesses	-0.5	3.5	0.0
injury_claim	-6220.0	21820.0	0.0
property_claim	-5215.0	20545.0	6.0
vehicle_claim	-502.5	81617.5	0.0

[99]:

```
# Cap outliers based on IQR bounds
for col in ['umbrella_limit', 'property_claim', 'vehicle_claim']:
    Q1 = final_df[col].quantile(0.25)
    Q3 = final_df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Apply capping
    final_df[col] = np.where(final_df[col] < lower_bound, lower_bound,␣
  ↪final_df[col])
    final_df[col] = np.where(final_df[col] > upper_bound, upper_bound,␣
  ↪final_df[col])

print("Outliers have been capped to IQR bounds.")
```

```
Outliers have been capped to IQR bounds.
```

[100]:

```
# Checking for outliers again after capping
outliers_summary = {}

for col in ['umbrella_limit', 'property_claim', 'vehicle_claim']:
    Q1 = final_df[col].quantile(0.25)
    Q3 = final_df[col].quantile(0.75)
    IQR = Q3 - Q1
```

```
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers_count = final_df[(final_df[col] < lower_bound) | (final_df[col] >
 upper_bound)].shape[0]

    # Storing summary information
    outliers_summary[col] = {
        "Lower Bound": lower_bound,
        "Upper Bound": upper_bound,
        "Outliers Count": outliers_count
    }

# Displaying the outliers summary after capping
pd.DataFrame(outliers_summary).T
```

[100]:

	Lower Bound	Upper Bound	Outliers Count
umbrella_limit	0.0	0.0	0.0
property_claim	-5215.0	20545.0	0.0
vehicle_claim	-502.5	81617.5	0.0

[101]:

```
# Define num_df to include only numerical columns
num_df = final_df.select_dtypes(include=[np.number]).columns
```

[102]:

```
# Apply Min-Max Scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
final_df[num_df] = scaler.fit_transform(final_df[num_df])

print("Feature scaling (Min-Max) applied to numerical columns.")
```

```
Feature scaling (Min-Max) applied to numerical columns.
```

[103]:

```
# Check min and max after scaling
print(final_df[num_df].describe().loc[['min', 'max']])
```

```
     months_as_customer  policy_deductable  umbrella_limit  capital-gains  \
min                 0.0                0.0             0.0            0.0
max                 1.0                1.0             0.0            1.0

     capital-loss  incident_hour_of_the_day  number_of_vehicles_involved  \
min           0.0                       0.0                          0.0
max           1.0                       1.0                          1.0

     bodily_injuries  witnesses  injury_claim  …  \
min              0.0        0.0           0.0  …
max              1.0        1.0           1.0  …

     insured_relationship_unmarried  insured_relationship_wife  \
```

```
min                              0.0                        0.0
max                              1.0                        1.0

     incident_type_Parked Car  incident_type_Single Vehicle Collision  \
min                       0.0                                     0.0
max                       1.0                                     1.0

     incident_type_Vehicle Theft  collision_type_Rear Collision  \
min                          0.0                            0.0
max                          1.0                            1.0

     collision_type_Side Collision  authorities_contacted_Fire  \
min                            0.0                         0.0
max                            1.0                         1.0

     authorities_contacted_Other  authorities_contacted_Police
min                          0.0                           0.0
max                          1.0                           1.0

[2 rows x 45 columns]
```

Summary: Ready for Export and Next Steps

Alright, our data is now fully prepped! We've taken care of:

- **Handling missing values**: Replaced missing values to ensure consistency.
- **Encoding categorical variables**: Converted categorical data to numerical format so models can work with it.
- **Outlier treatment**: Capped extreme values to keep our data balanced.
- **Feature scaling**: Standardized the numerical columns to a [0, 1] range for smoother model performance.

Now we're ready to export this cleaned and processed DataFrame to a CSV file. This exported file will be our finalized dataset, and here's what we can do with it next:

1. **Training and Testing**: Use this file to split the data into training and test sets for building and evaluating our machine learning models.
2. **Deployment**: Since the data is clean, standardized, and model-ready, we can also use this same file in deployment for real-world predictions.

[104]:
```
data.head()
```

[104]:
```
   months_as_customer policy_csl  policy_deductable  policy_annual_premium  \
0                 328    250/500               1000                1406.91
1                 228    250/500               2000                1197.22
2                 134    100/300               2000                1413.14
3                 256    250/500               2000                1415.74
4                 228   500/1000               1000                1583.91

   umbrella_limit insured_sex insured_education_level insured_occupation  \
```

```
0          0    MALE          MD      craft-repair
1    5000000    MALE          MD  machine-op-inspct
2    5000000  FEMALE         PhD             sales
3    6000000  FEMALE         PhD      armed-forces
4    6000000    MALE   Associate             sales

  insured_relationship  capital-gains  …  incident_hour_of_the_day  \
0              husband          53300  …                         5
1       other-relative              0  …                         8
2            own-child          35100  …                         7
3            unmarried          48900  …                         5
4            unmarried          66000  …                        20

  number_of_vehicles_involved property_damage bodily_injuries witnesses  \
0                           1             YES               1         2
1                           1              NO               0         0
2                           3              NO               2         3
3                           1              NO               1         2
4                           1              NO               0         1

   police_report_available  injury_claim property_claim  vehicle_claim  \
0                      YES          6510          13020          52080
1                       NO           780            780           3510
2                       NO          7700           3850          23100
3                       NO          6340           6340          50720
4                       NO          1300            650           4550

   fraud_reported
0               Y
1               Y
2               N
3               Y
4               N

[5 rows x 25 columns]
```

[105]:

```python
# Exporting the final cleaned and preprocessed data to a CSV file
final_df.to_csv('insuranceFraud_final_processed_data.csv', index=False)
print("Data exported to 'insuranceFraud_final_processed_data.csv' successfully!
 ↪")
```

```
Data exported to 'insuranceFraud_final_processed_data.csv' successfully!
```

[ ]:

```python

```