

leptospirosis-detection-rf

June 24, 2024

1 Leptospirosis Detection

1.1 Load train dataset

```
[1]: import pandas as pd
import numpy as np

train_csv = pd.read_csv('train.csv').drop('ID', axis = 1)
train_data = pd.DataFrame(train_csv)
```

C:\Users\Lasani\AppData\Local\Temp\ipykernel_18576\2849540320.py:4:
DtypeWarning: Columns (494,597,599,600,601,603,604) have mixed types. Specify
dtype option on import or set low_memory=False.

```
train_csv = pd.read_csv('train.csv').drop('ID', axis = 1)
```

```
[2]: '''#Adjusting display options in the console
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)'''
```

```
[2]: "#Adjusting display options in the console\npd.set_option('display.max_columns',
None)\npd.set_option('display.max_rows', None)"
```

1.2 Explore train data

```
[3]: #Get number of rows and columns
train_data.shape
```

```
[3]: (1387, 805)
```

```
[4]: #Explore the first few rows of the dataset
train_data.head()
```

```
[4]:   Year  Month  Hospital  Sample  ICU  OPD  Sex  Age  Ethnicity  Income  ...  \
0  2018    11         7        1     2    2    2   53          1   35000  ...
1  2018     1         7        1     2    2    1   17          1     99  ...
2  2018     5         7        1     2    2    1   47          1  40000  ...
3  2018     1         7        1     2    2    1   21          1  30000  ...
4  2016     8         7        1     2    1    1   99          1     99  ...
```

	FU_L.interrogansserovarMankarsostr.Mankarso	\
0	NaN	
1	NaN	
2	NaN	
3	NaN	
4	NaN	

	FU_L.santarosaiserovarGeorgiastr.LT117	\
0	NaN	
1	NaN	
2	NaN	
3	NaN	
4	NaN	

	FU_L.santarosaiserovarPyrogenesstr.Salinem	\
0	NaN	
1	NaN	
2	NaN	
3	NaN	
4	NaN	

	FU_L.interrogansserovarBataviaastr.VanTienan	\
0	NaN	
1	NaN	
2	NaN	
3	NaN	
4	NaN	

	FU_L.interrogansserovarAlexistr.616	\
0	NaN	
1	NaN	
2	NaN	
3	NaN	
4	NaN	

	FU_L.interrogansserovarAustralisstr.Ballico	\
0	NaN	
1	NaN	
2	NaN	
3	NaN	
4	NaN	

	FU_L.interrogansserovarwolfiistr.3705	FU_L.interrogansserovarWeerasinghe	\
0	NaN	NaN	
1	NaN	NaN	
2	NaN	NaN	

3	NaN	NaN
4	NaN	NaN

	FU_Patoc	Final
0	NaN	2
1	NaN	1
2	NaN	2
3	NaN	2
4	NaN	2

[5 rows x 805 columns]

```
[5]: #Get data types of columns
train_data.dtypes
```

```
[5]: Year                                int64
Month                                int64
Hospital                            int64
Sample                             int64
ICU                                int64
...
FU_L.interrogansserovarAustralisstr.Ballico    float64
FU_L.interrogansserovarwolffiistr.3705         float64
FU_L.interrogansserovarWeerasinghe             float64
FU_Patoc                                       float64
Final                                         int64
Length: 805, dtype: object
```

1.3 Handle duplicates

```
[6]: #Check for duplicates
duplicates = train_data.duplicated().sum()
print("Number of duplicate rows: ", duplicates)
```

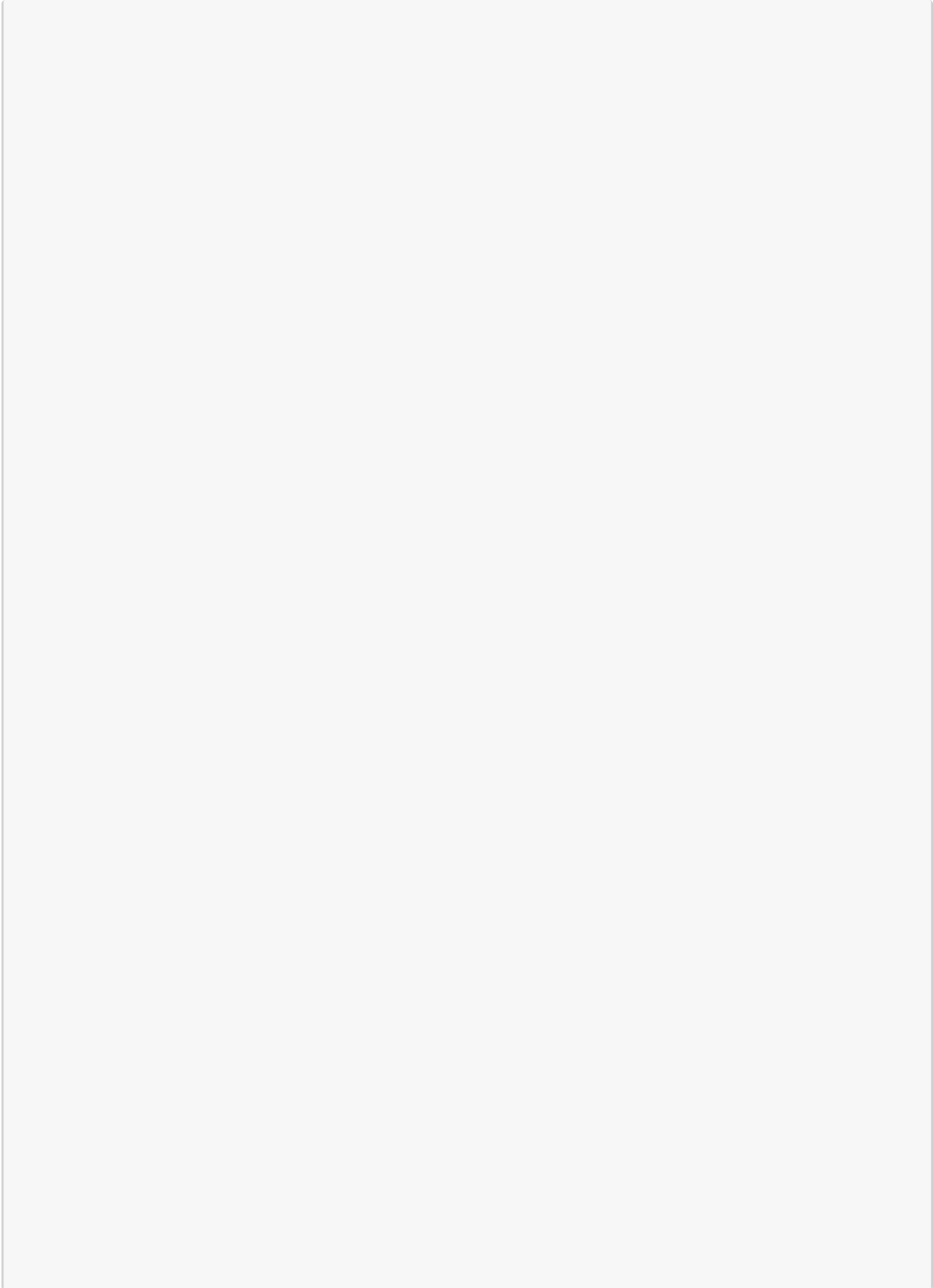
Number of duplicate rows: 82

```
[7]: #Remove duplicates
train_data = train_data.drop_duplicates()
print('Records after removing duplicates: ', train_data.shape[0])
```

Records after removing duplicates: 1305

1.4 Drop unnecessary (repeated) columns

[8]:



```
train_data.drop(columns = columns_to_drop, inplace = True)
train_data.shape
```

[8]: (1305, 219)

1.5 Replace vague values of numerical columns

```
[9]: #Replace values starting with 'fiel' or 'occ' with 99
train_data = train_data.map(lambda x: 99 if isinstance(x, str) and (x.lower().
    ↳startswith('fiel') or x.lower().startswith('occ')) else x)
print(train_data)
```

	Hospital	Sample	ICU	OPD	Sex	Age	Prophylactics	Pasttreatments	\
0	7	1	2	2	2	53	3	1	
1	7	1	2	2	1	17	2	1	
2	7	1	2	2	1	47	2	1	
3	7	1	2	2	1	21	2	1	
4	7	1	2	1	1	99	2	1	
...	
1382	4	2	2	2	1	62	99	99	
1383	7	1	2	2	2	59	2	1	
1384	5	1	2	1	1	56	2	1	
1385	4	2	2	2	1	61	99	99	
1386	8	1	2	2	1	36	2	1	

	Pastantibiotics	Chronicillness	...	S.amylase10	S.phosphate10	ALP10	\
0	1		2	...	99	99	99.0
1	1		2	...	99	99	99.0
2	2		2	...	99	99	99.0
3	1		2	...	99	99	99.0
4	3		99	...	99	99	99.0
...	
1382	99		99	...	99	99	99.0
1383	1		2	...	99	99	99.0
1384	3		1	...	99	99	99.0
1385	99		99	...	99	99	99.0
1386	3		2	...	99	99	99.0

	WPqPCRDiagnosis	UrineqPCRDiagnosis	CultureqPCRDia	SerumqPCRDiagnosis	\
0	3		99	99	99
1	1		3	3	99
2	3		3	3	99
3	3		3	3	99
4	3		3	99	3
...	
1382	3		99	99	99
1383	3		99	99	99

1384	3	99	99	99
1385	3	99	99	99
1386	3	2	99	99

	UFqPCRDia	Isolate	Final
0	99	2	2
1	99	2	1
2	99	2	2
3	99	2	2
4	99	98	2
...
1382	99	98	2
1383	99	98	2
1384	99	98	2
1385	99	2	2
1386	3	2	1

[1305 rows x 219 columns]

1.6 Drop missing values of more than 80%

```
[10]: #Convert values called '99' to missing values as in the description
train_data.replace(99, pd.NA, inplace = True)
train_data.replace('99', pd.NA, inplace = True)

print(train_data.head())
```

	Hospital	Sample	ICU	OPD	Sex	Age	Prophylactics	Pasttreatments	\
0	7	1	2	2	2	53	3	1	
1	7	1	2	2	1	17	2	1	
2	7	1	2	2	1	47	2	1	
3	7	1	2	2	1	21	2	1	
4	7	1	2	1	1	<NA>	2	1	

	Pastantibiotics	Chronicillness	...	S.amylase10	S.phosphate10	ALP10	\
0	1		2	...	<NA>	<NA>	<NA>
1	1		2	...	<NA>	<NA>	<NA>
2	2		2	...	<NA>	<NA>	<NA>
3	1		2	...	<NA>	<NA>	<NA>
4	3		<NA>	...	<NA>	<NA>	<NA>

	WPqPCRDia	UrineqPCRDia	CultureqPCRDia	SerumqPCRDia	\
0	3	<NA>	<NA>	<NA>	
1	1	3	3	<NA>	
2	3	3	3	<NA>	
3	3	3	3	<NA>	
4	3	3	<NA>	3	

	UFqPCRDiag	Isolate	Final
0	<NA>	2	2
1	<NA>	2	1
2	<NA>	2	2
3	<NA>	2	2
4	<NA>	98	2

[5 rows x 219 columns]

```
[11]: #Identify missing value counts
missing_counts = train_data.isnull().sum()
print(missing_counts)
```

```
Hospital          0
Sample            0
ICU               20
OPD               20
Sex               67
...
CultureqPCRDia    1040
SerumqPCRDiagnosis 1023
UFqPCRDiag        1147
Isolate           0
Final             0
Length: 219, dtype: int64
```

```
[12]: #Drop columns with too many missing values
missing_percentages = missing_counts / len(train_data)
threshold = 0.85
columns_to_drop = missing_percentages[missing_percentages > threshold].index
train_data.drop(columns = columns_to_drop, inplace = True)

#Verification
train_data.shape
```

[12]: (1305, 176)

1.7 Divide variables into categorical and numerical

```
[13]: #Group variables into categorical and numerical by defining a threshold value
      ↳ for number of unique values in each column
categorical_vars = []
numerical_vars = []
unique_threshold = 12

for column in train_data.columns:
    unique_count = train_data[column].nunique()
```



```

if unique_count <= unique_threshold:
    categorical_vars.append(column)
else:
    numerical_vars.append(column)

```

```

[14]: #Adjust after manually going through variable data types
categorical_vars.remove('Puscells')
numerical_vars.append('Puscells')

print("After manual adjustment:")
print("Categorical variables:", categorical_vars, "\n")
print("Numerical variables:", numerical_vars)

```

After manual adjustment:

```

Categorical variables: ['Hospital', 'Sample', 'ICU', 'OPD', 'Sex',
'Prophylactics', 'Pasttreatments', 'Pastantibiotics', 'Chronicillness',
'Possibleexposure', 'Usualdrinkingwatersource', 'Usualbathingwatersource',
'Sourceofwaterforhousehold', 'Accumilationofrefusal',
'Availabilityofpublicgarbagecollectionprocedure',
'HomeStreamrivercanaloranyotherrunningwatersource',
'WorkplaceStreamrivercanaloranyotherrunningwatersource',
'Homepondlaketankoranyotherstagnantwatersource',
'WorkPlacepondlaketankoranyotherstagnantwatersource', 'Homemarshywetland',
'Workplacemarshywet', 'HomeBushes', 'Workplacebushes', 'Homeforest',
'Workplaceforest', 'Homeworkingpaddyfield', 'WorkPlaceworkingpaddyfield',
'Homeabondantpaddyfield', 'Workplaceabondantpaddyfield',
'Homeotheragricultural', 'workplaceotheragricultural', 'Homeanimalfarm',
'Workplaceanimalfarm', 'Homegarbageaccumilation', 'Homeblockeddrainage',
'Homesewer', 'Rathome', 'RatWorkplace', 'RatNeighbourhood', 'Cattlehome',
'CattleWorkplace', 'Urumeeyahome', 'Urumeeyaworkplace', 'UrimeeyaNeighbourhood',
'OtherrhodentsHome', 'Marshlandexposure', 'Wetsoilexposure', 'Floodexposure',
'Forestexposure', 'BushesExposure', 'Otheragriexposure',
'Naturalrunningwaterexposure', 'Stagnantwaterexposure',
'Manmaderunningwaterexposure', 'Drainsexposure', 'Paddyfieldexposure',
'Walkingbarefootoutdoor', 'CattleHandle', 'BuffaloHandle', 'Goathandle',
'Pighandle', 'Cathandle', 'Doghandle', 'Feveronset', 'Headacheonset',
'Musclepainonset', 'Cnsuffusiononset', 'Jaundiceonset', 'Skinrashonset',
'Oliguriaonset', 'Anuriaonset', 'SOBonset', 'Coughonset', 'Haemoptasisonset',
'Chestpainonset', 'Nauseaonset', 'Vomitingonset', 'Diarrhoeaonset',
'Bleedingonset', 'Mucosalrashonset', 'Prostrationonset', 'Rigorsonset',
'Photophobiaonset', 'Chillsonset', 'Muscletendernessonset',
'Psychoticsymptomsonset', 'Confusiononset', 'Feverad', 'Headachead', 'Chillsad',
'Rigorsad', 'Musclepainad', 'Muscletendernessad', 'Nauseaad',
'Vomitingadmission', 'Cnsuffusionad', 'Skinrashad', 'Mucosalrashad',
'Prostrationad', 'Diarrhoeaad', 'OliguriaAd', 'Anuriaad', 'Jaundicead',
'Hepatic tendernessad', 'Hepatomegalyad', 'Spleenimegalyad', 'Lymphadenopathyad',
'Photophobiaad', 'Neckstiffnessad', 'Psychoticsymptomsad', 'Confusionad',

```

```
'Coughad', 'Haemoptasisad', 'SOBadd', 'Chestpainad', 'Bleedingad', 'Headache10',
'Fever10', 'Chills10', 'Rigors10', 'Musclepain10', 'Mustender10', 'Nausea10',
'Vomiting10', 'Consuf10', 'Skinrash10', 'Mucorash10', 'Prostration10',
'diarrhea10', 'Oliguria10', 'Anuria10', 'Jaundice10', 'hepatictender10',
'hepatomegaly10', 'Spleenomegaly10', 'Lymphadenopathy10', 'Photophobia10',
'Neckstiffness10', 'Confusion10', 'Cough10', 'Haemoptysis10', 'SOB10',
'Chestpain10', 'Bleeding10', 'Albumin', 'WPqPCRDiagnosis', 'UrineqPCRDiagnosis',
'CultureqPCRDia', 'SerumqPCRDiagnosis', 'Isolate', 'Final']
```

```
Numerical variables: ['Age', 'PRad', 'SBPadd', 'DBPadd', 'WBCcount', 'Ncount',
'N', 'Lcount', 'L', 'Plateletcount', 'PCV', 'RBC', 'CRP', 'ESR', 'Redcells',
'Na', 'K', 'AST', 'ALT', 'T.Bilirubin', 'D.Bilirubin', 'S.creatinine', 'B.urea',
'ALP', 'Puscells']
```

```
[15]: #Adjust data types in numerical columns
train_data[numerical_vars] = train_data[numerical_vars].apply(pd.to_numeric,
↳errors = 'coerce')

#Verify data types
train_data.dtypes
```

```
[15]: Hospital          int64
Sample              int64
ICU                 object
OPD                 object
Sex                 object
...
UrineqPCRDiagnosis  object
CultureqPCRDia      object
SerumqPCRDiagnosis  object
Isolate             int64
Final               int64
Length: 176, dtype: object
```

1.8 Drop highly correlated numerical columns

```
[16]: #Identify highly correlated numerical columns
corr_matrix = train_data[numerical_vars].corr()
#Apply a threshold to get the most correlated ones
threshold = 0.8
highly_correlated = corr_matrix.abs() > threshold
highly_correlated_columns = highly_correlated.any(axis = 0)

#Determine highly correlated pairs
correlated_pairs = []
for i in range(len(corr_matrix.columns)):
    for j in range(i + 1, len(corr_matrix.columns)):
```

```

        if abs(corr_matrix.iloc[i, j]) > threshold:
            correlated_pairs.append((corr_matrix.columns[i], corr_matrix.
↪columns[j]))

print("Highly correlated column pairs:")
print(correlated_pairs)

```

Highly correlated column pairs:

```
[('SBPadd', 'DBPadd'), ('AST', 'ALT'), ('T.Bilirubin', 'D.Bilirubin')]
```

```

[17]: #Drop one of highly correlated variables from pairs to avoid multicollinearity
train_data = train_data.drop(columns = ['DBPadd', 'ALT', 'D.Bilirubin'])

numerical_vars.remove('DBPadd')
numerical_vars.remove('ALT')
numerical_vars.remove('D.Bilirubin')

train_data.shape

```

```
[17]: (1305, 173)
```

1.9 Check for outliers through boxplots in numerical data

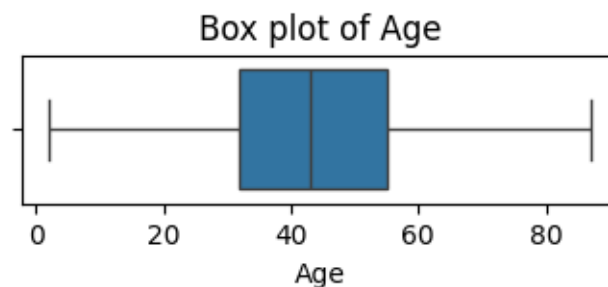
```

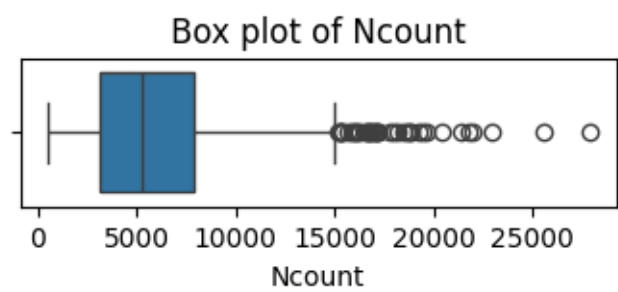
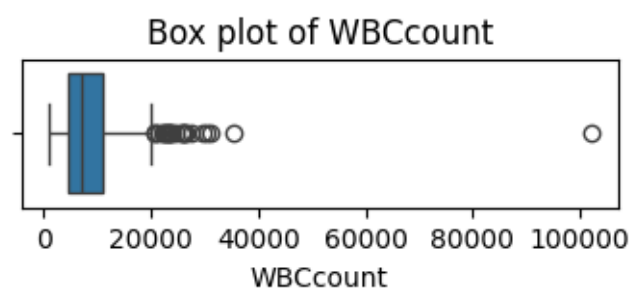
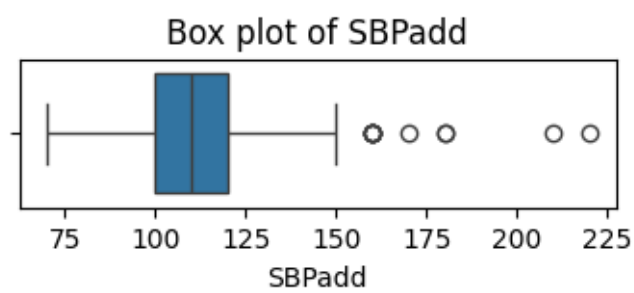
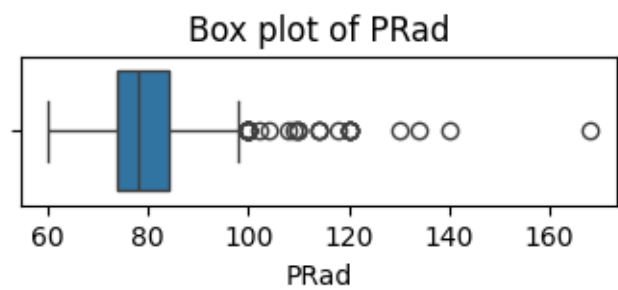
[18]: import matplotlib.pyplot as plt
import seaborn as sns

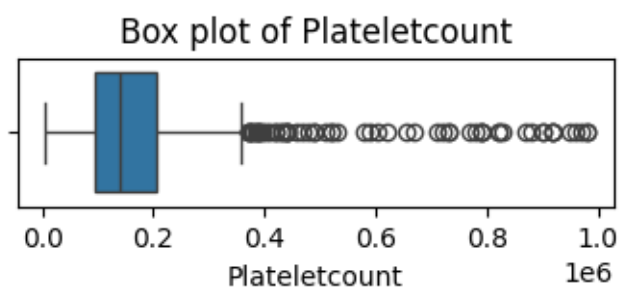
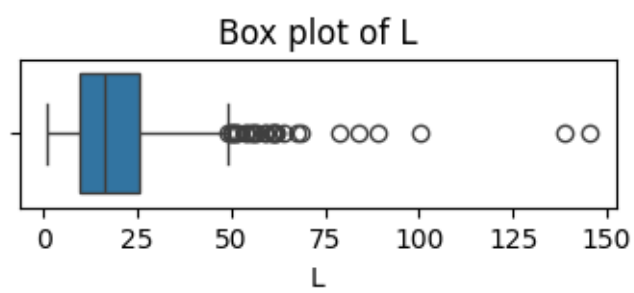
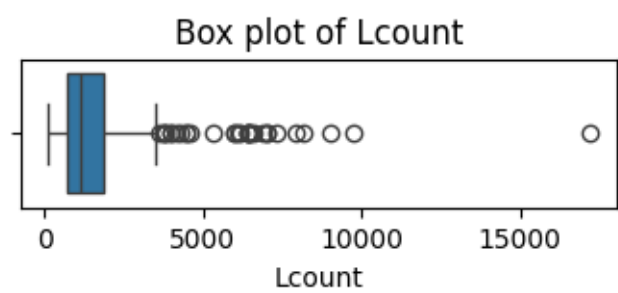
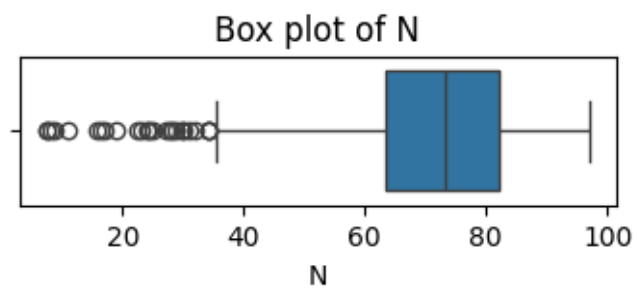
for col in numerical_vars:
    plt.figure(figsize = (4, 1))
    sns.boxplot(x = train_data[col])
    plt.title(f'Box plot of {col}')
    plt.xlabel(col)
    plt.show()

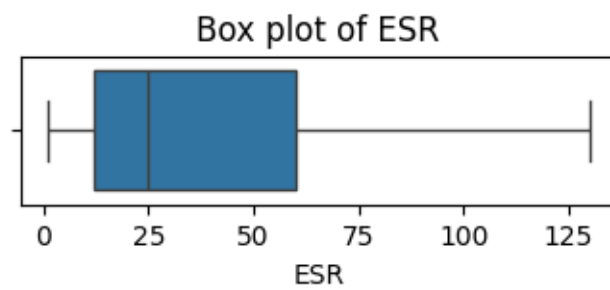
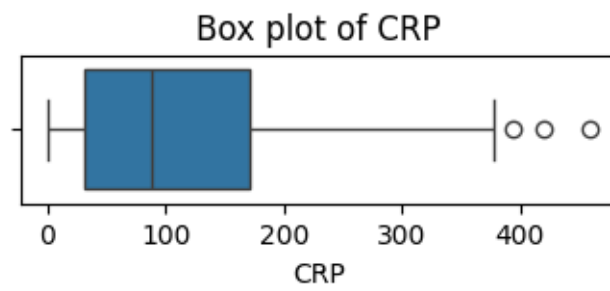
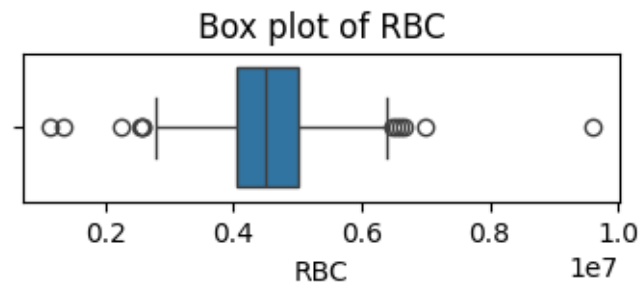
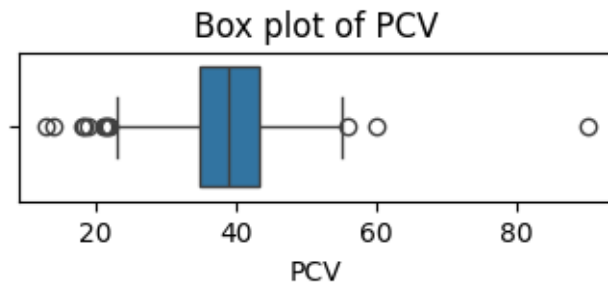
#Detecting heavy outliers in most of the columns

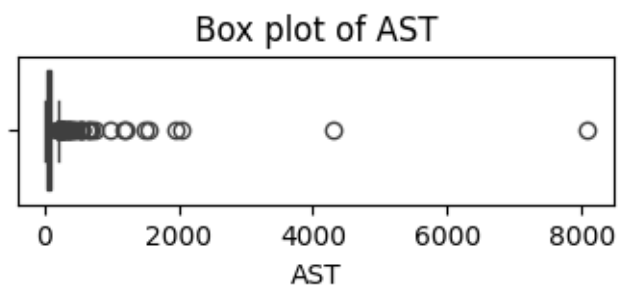
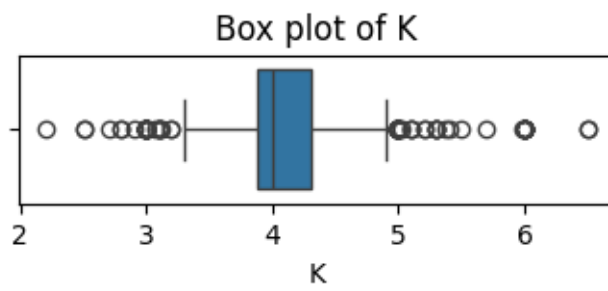
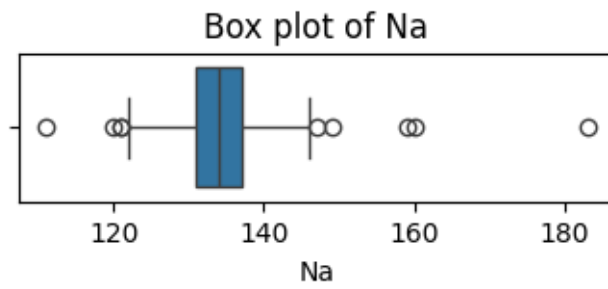
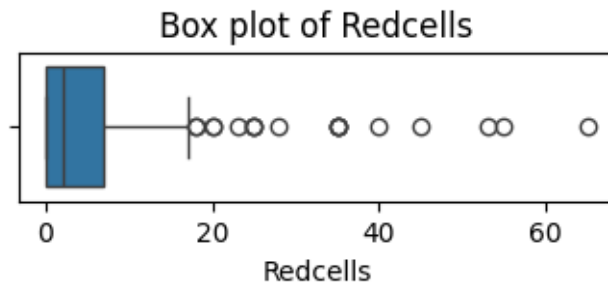
```



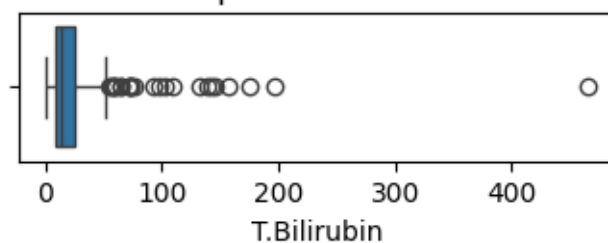




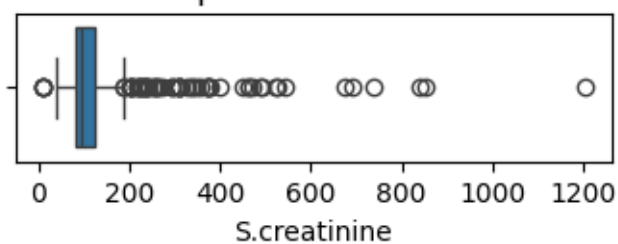




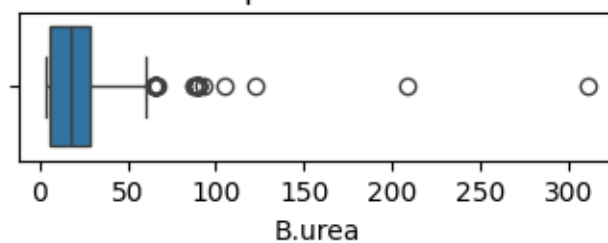
Box plot of T.Bilirubin



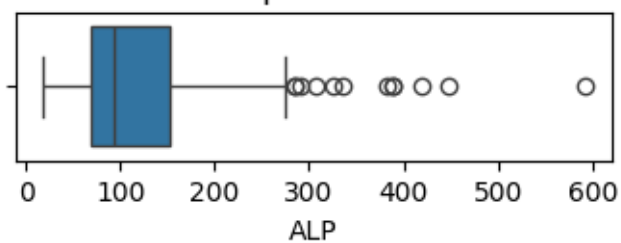
Box plot of S.creatinine

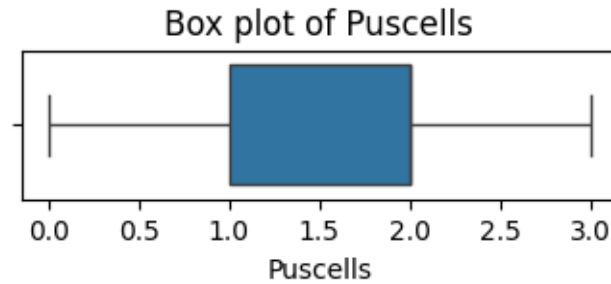


Box plot of B.urea



Box plot of ALP





1.10 Impute numerical variables

```
[19]: #With median
for col in numerical_vars:
    if train_data[col].isnull().sum() > 0:
        median_val = train_data[col].median()
        train_data[col].fillna(median_val, inplace = True)
```

1.11 Perform robust scaling on numerical data

```
[20]: from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()
train_data[numerical_vars] = scaler.fit_transform(train_data[numerical_vars])
train_data[numerical_vars] = pd.DataFrame(train_data[numerical_vars], columns =
↳ numerical_vars)
train_data[numerical_vars]
```

```
[20]:
```

	Age	PRad	SBPadd	WBCcount	Ncount	N	Lcount	\
0	0.476190	8.0	10.0	1.670487	4.675926	1.201724	3.878505	
1	-1.238095	-2.0	0.0	-1.289398	0.000000	0.000000	0.000000	
2	0.190476	10.0	-10.0	0.226361	0.712963	0.453955	0.000000	
3	-1.047619	0.0	0.0	-0.744986	-2.925926	-5.645010	4.018692	
4	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	
...	
1382	0.904762	0.0	0.0	0.000000	0.000000	0.000000	0.000000	
1383	0.761905	22.0	-6.0	-0.309456	0.000000	0.000000	0.000000	
1384	0.619048	0.0	0.0	0.000000	0.000000	0.000000	0.000000	
1385	0.857143	0.0	0.0	0.000000	0.000000	0.000000	0.000000	
1386	-0.333333	0.0	10.0	-0.243553	0.000000	1.960496	-2.196262	

	L	Plateletcount	PCV	...	ESR	Redcells	Na	K	AST	\
0	-0.286085	-0.964286	-8.60	...	0.0	0.0	-1.0	-0.8	5.0	
1	0.000000	0.000000	0.00	...	0.0	-1.0	0.0	0.0	0.0	

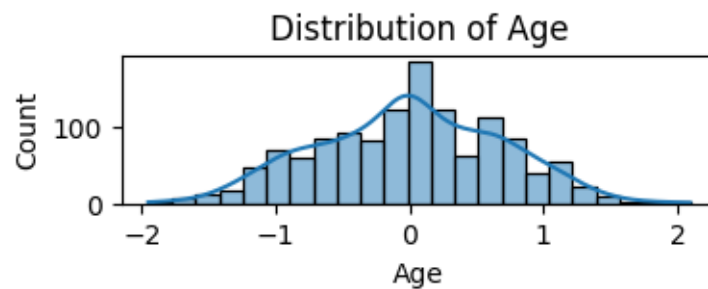
2	-0.469785	-1.250000	-1.70	...	0.0	0.0	-5.0	0.0	0.0
3	5.834799	-0.821429	-1.30	...	0.0	0.0	0.0	0.0	0.0
4	0.000000	0.000000	0.00	...	0.0	0.0	0.0	0.0	0.0
...
1382	0.000000	0.000000	0.00	...	0.0	0.0	0.0	0.0	0.0
1383	0.000000	0.000000	0.00	...	0.0	0.0	0.0	0.0	0.0
1384	0.000000	0.000000	0.00	...	0.0	0.0	0.0	0.0	0.0
1385	0.000000	0.000000	0.00	...	0.0	0.0	0.0	0.0	0.0
1386	-1.271963	-0.964286	-0.85	...	0.0	0.0	0.0	0.0	0.0

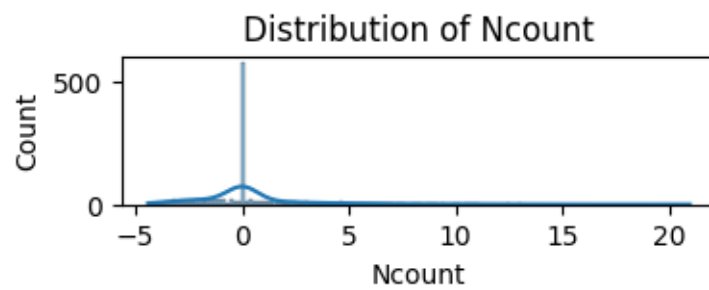
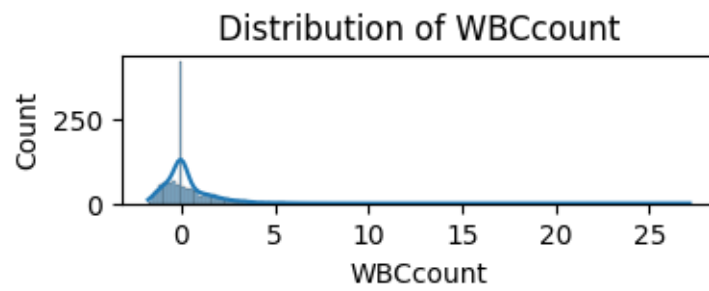
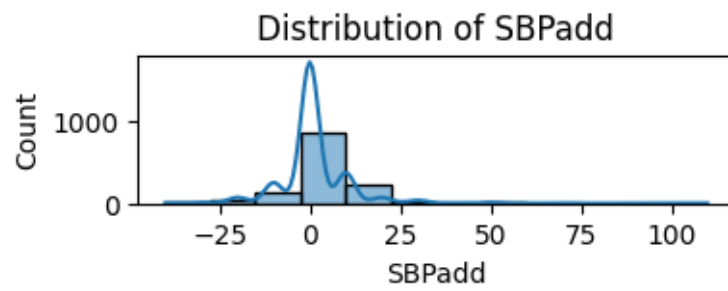
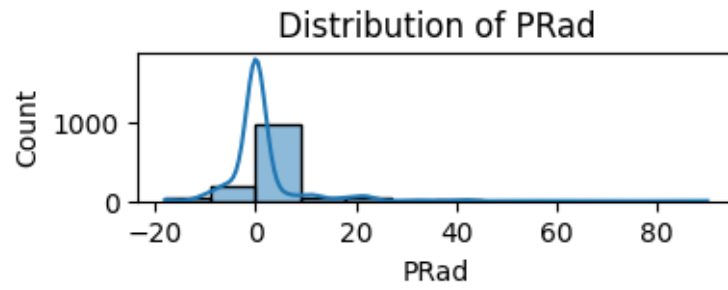
	T.Bilirubin	S.creatinine	B.urea	ALP	Puscells
0	25.3	197.0	4.2	114.0	0.0
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0
...
1382	0.0	0.0	0.0	0.0	0.0
1383	0.0	0.0	0.0	0.0	0.0
1384	0.0	0.0	0.0	0.0	0.0
1385	0.0	0.0	0.0	0.0	0.0
1386	0.0	0.0	0.0	0.0	0.0

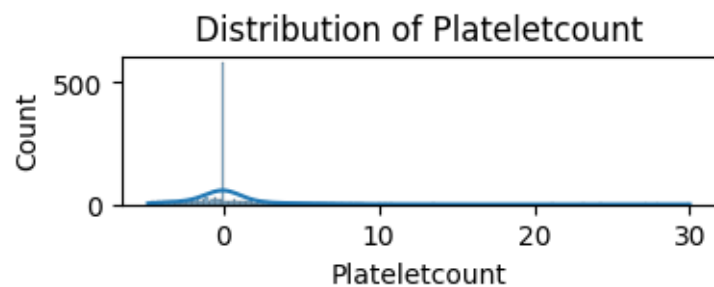
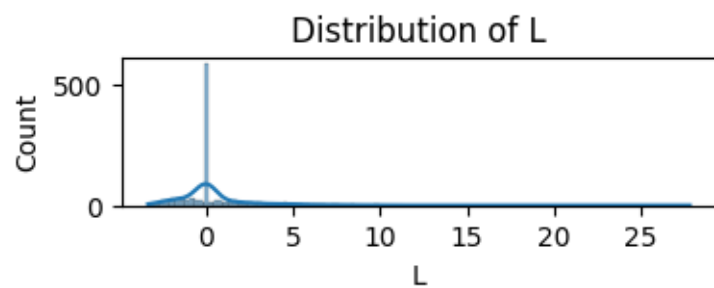
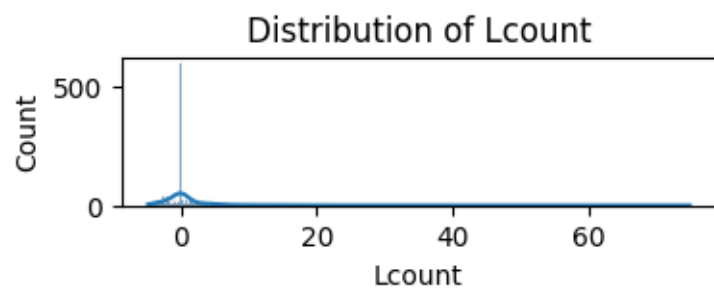
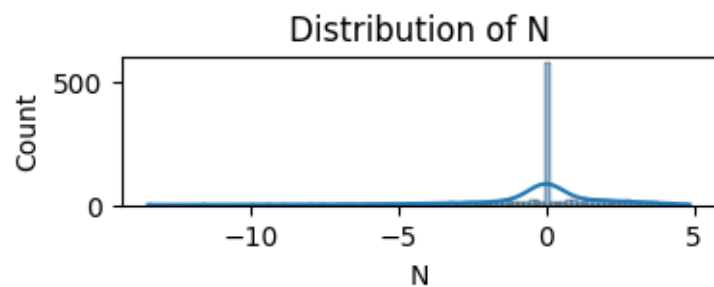
[1305 rows x 22 columns]

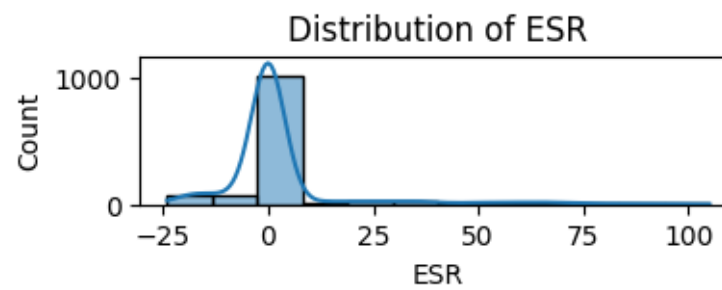
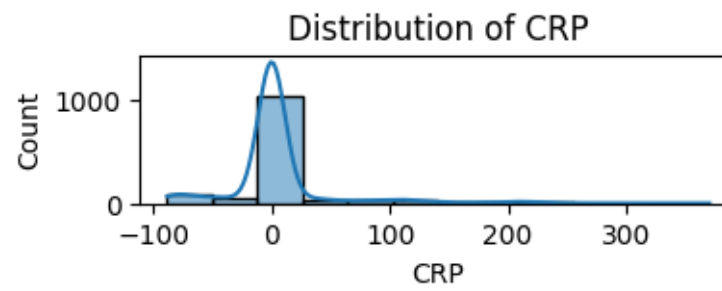
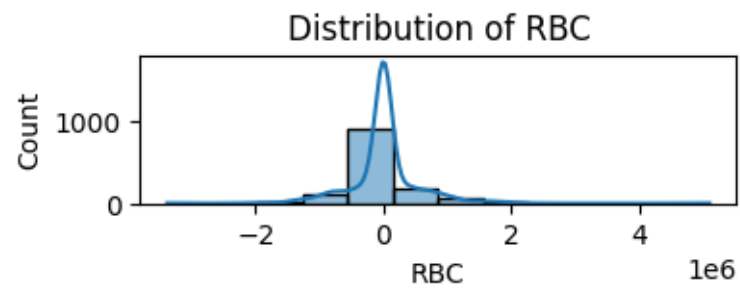
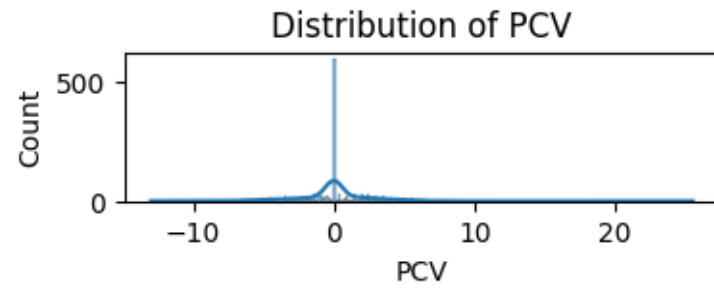
1.12 EDA on numerical data

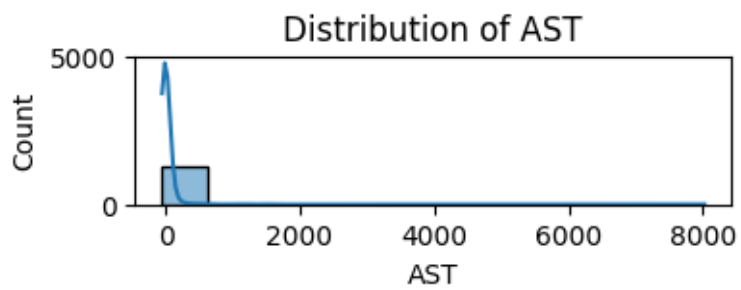
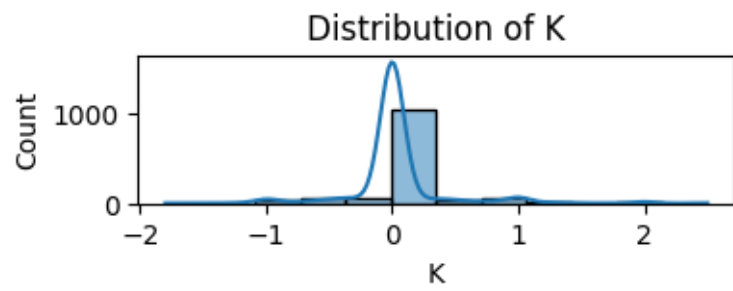
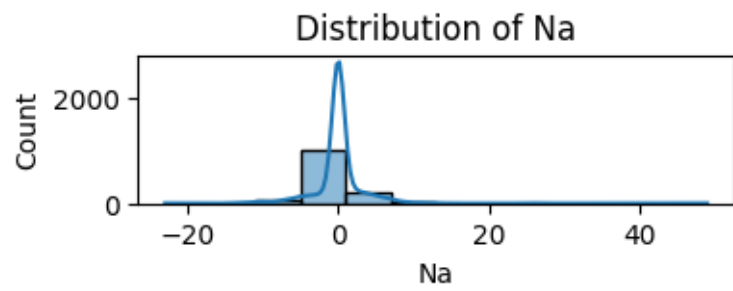
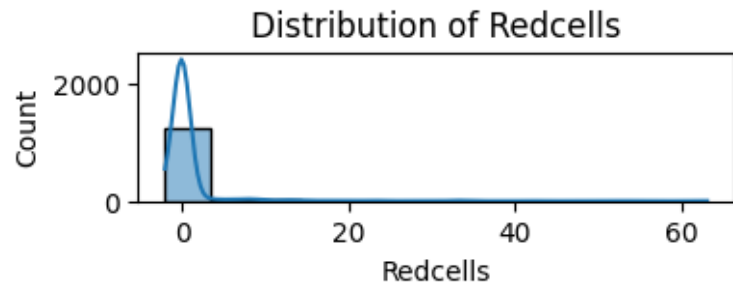
```
[21]: for column in train_data[numerical_vars]:
plt.figure(figsize = (4, 1))
sns.histplot(train_data[column], kde = True)
plt.title(f'Distribution of {column}')
plt.show()
```

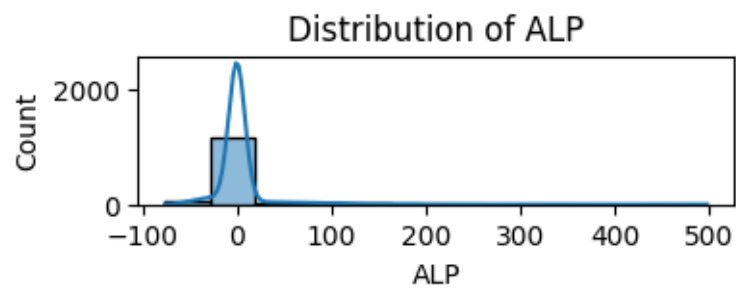
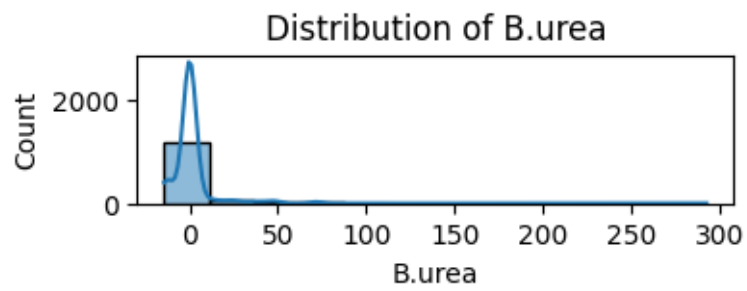
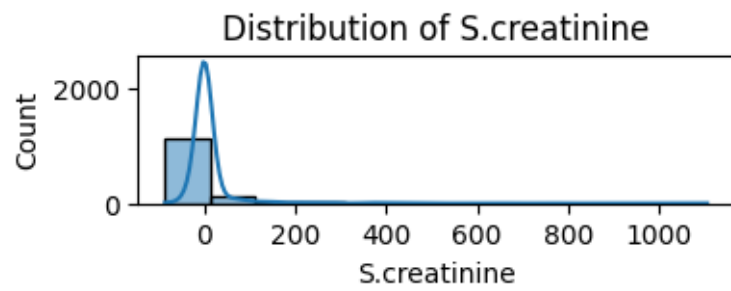
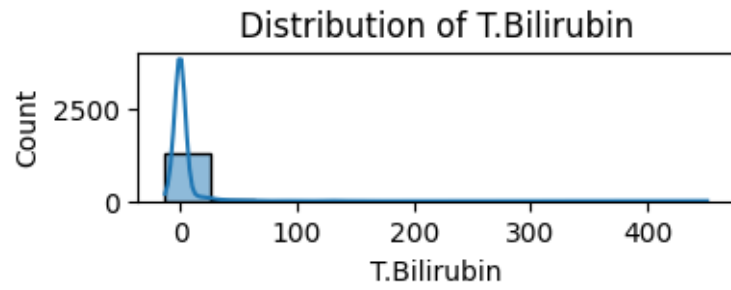


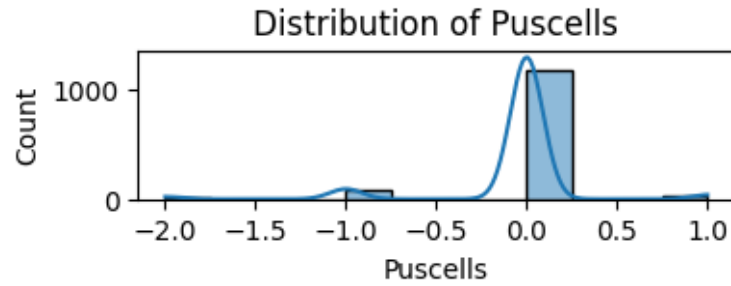












1.13 Impute categorical variables

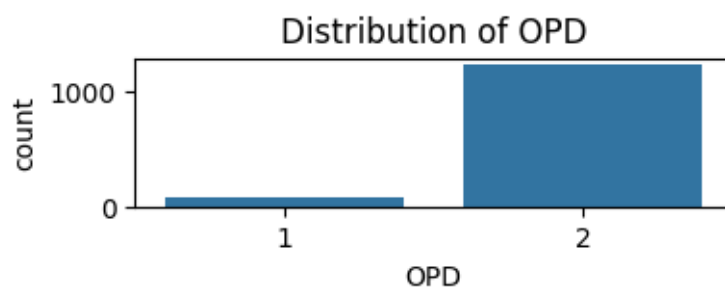
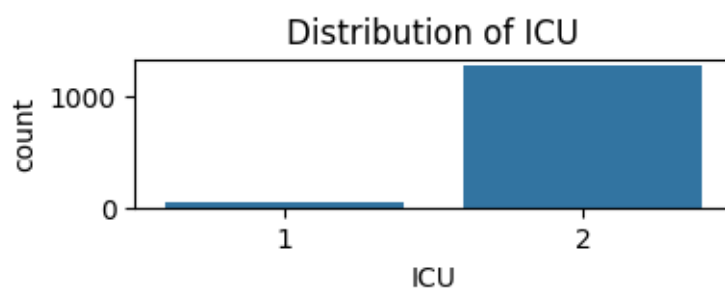
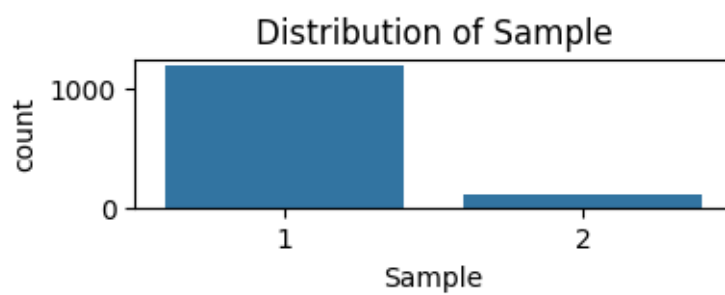
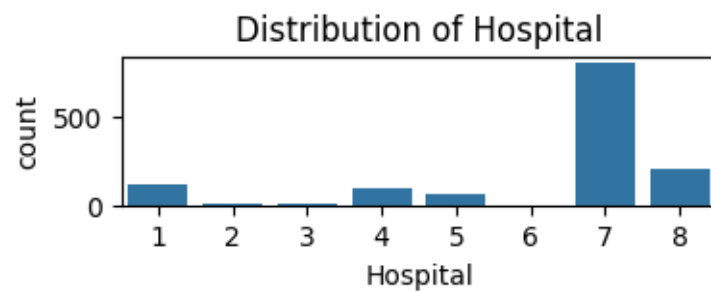
```
[22]: #With mode
      for col in categorical_vars:
          train_data[col].fillna(train_data[col].mode()[0], inplace = True)
```

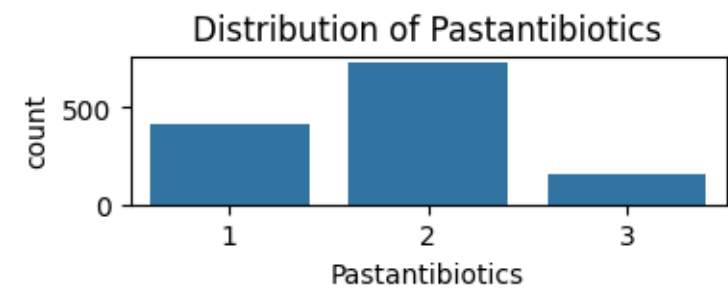
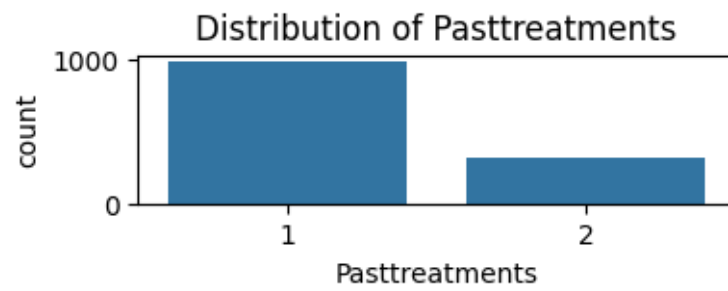
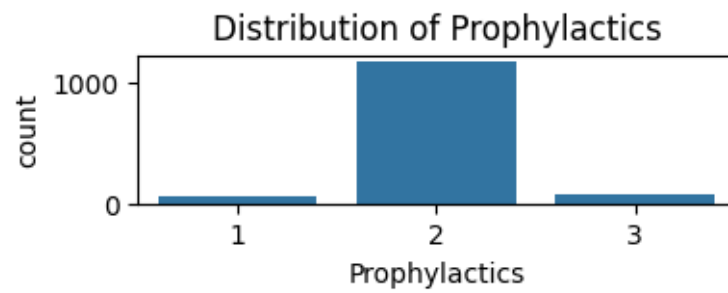
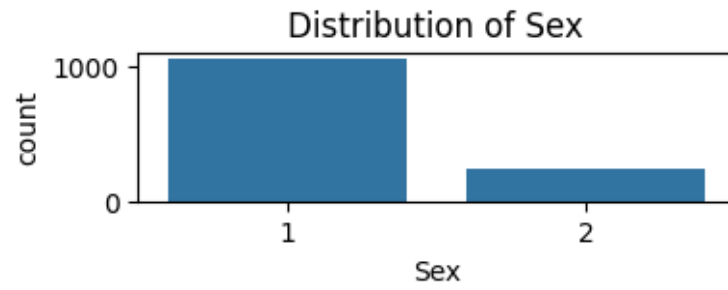
```
[23]: #Verification
      print(train_data.isnull().sum())
```

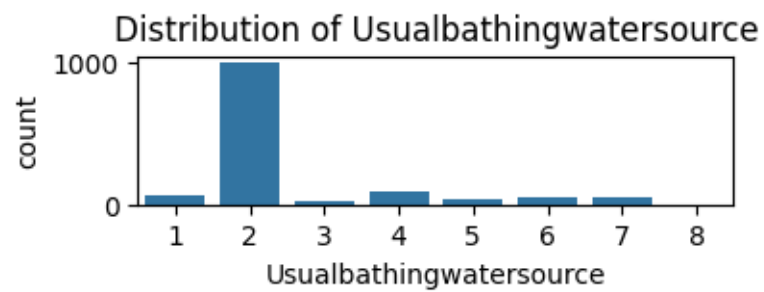
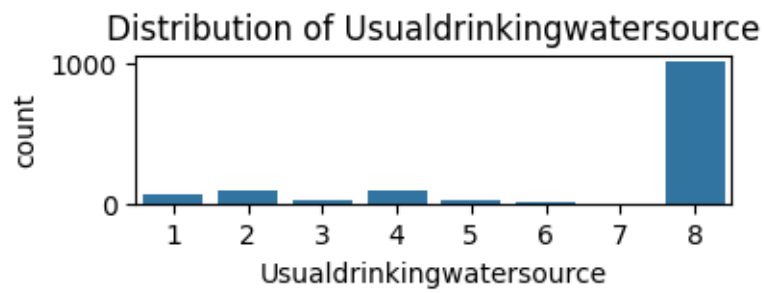
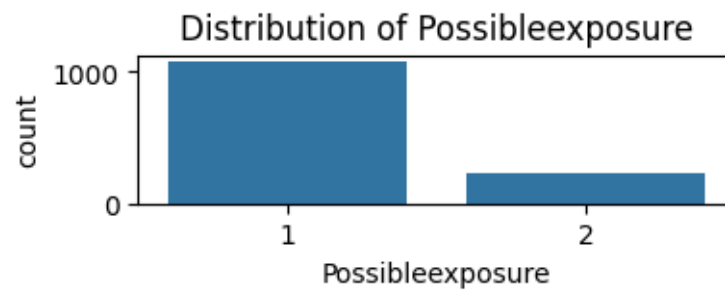
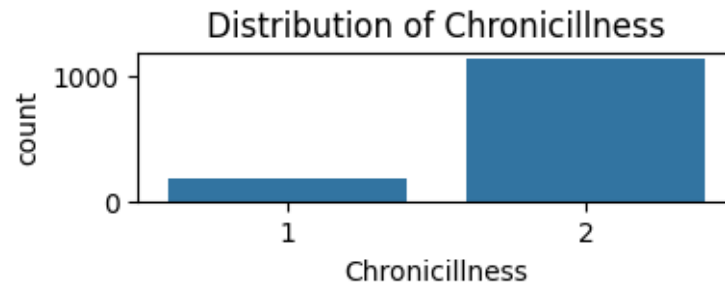
```
Hospital          0
Sample            0
ICU               0
OPD               0
Sex               0
..
UrineqPCRDiaosis  0
CultureqPCRDia    0
SerumqPCRDiaosis  0
Isolate           0
Final             0
Length: 173, dtype: int64
```

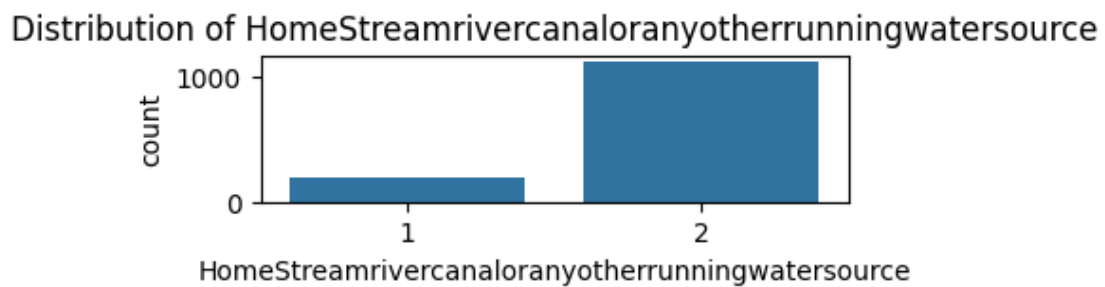
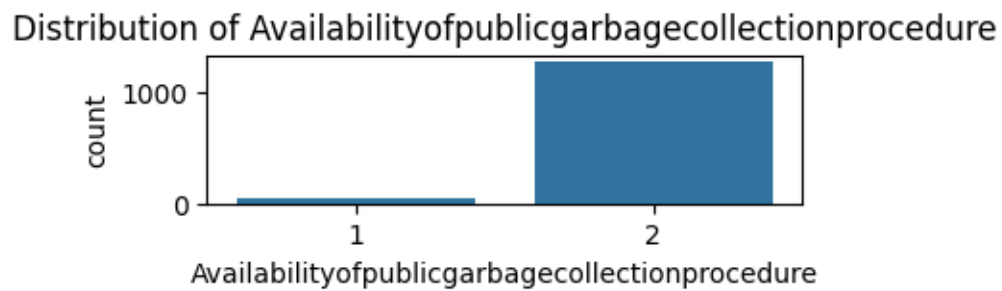
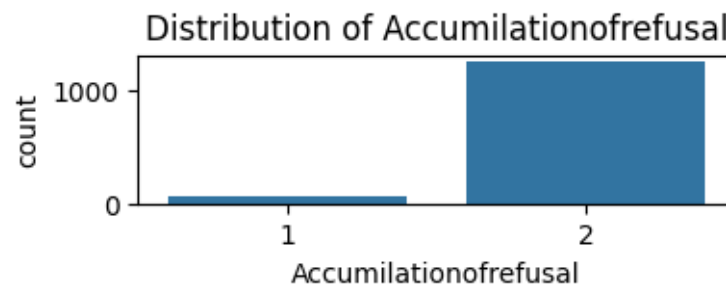
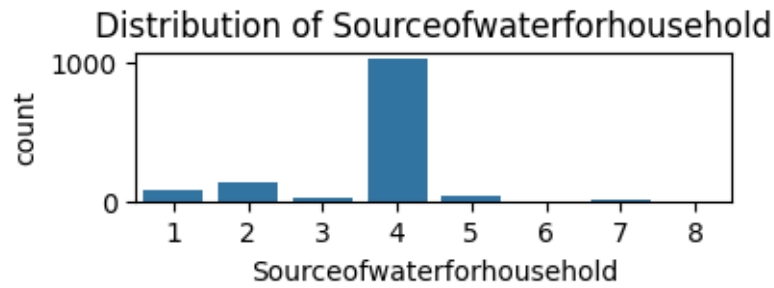
1.14 EDA on categorical data

```
[24]: for column in categorical_vars:
      plt.figure(figsize = (4, 1))
      sns.countplot(x = train_data[column])
      plt.title(f'Distribution of {column}')
      plt.show()
```

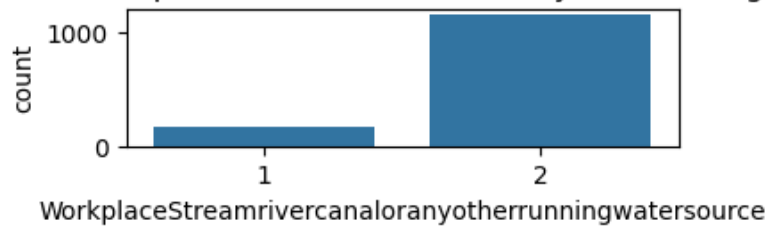





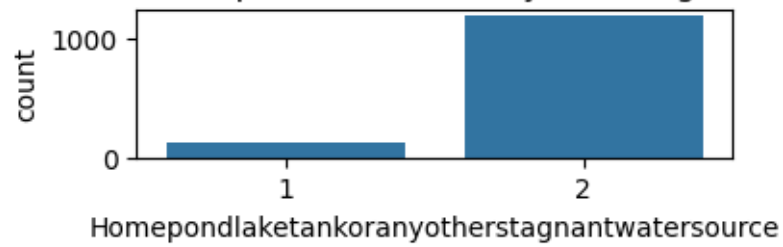




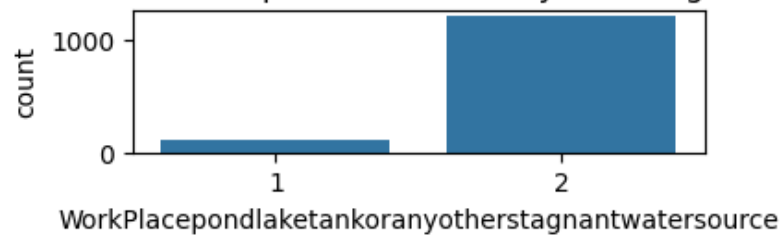
Distribution of WorkplaceStreamrivercanaloranyotherrunningwatersource



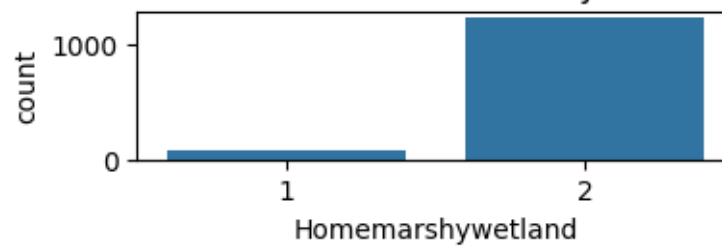
Distribution of Homepondlaketankoranyotherstagnantwatersource

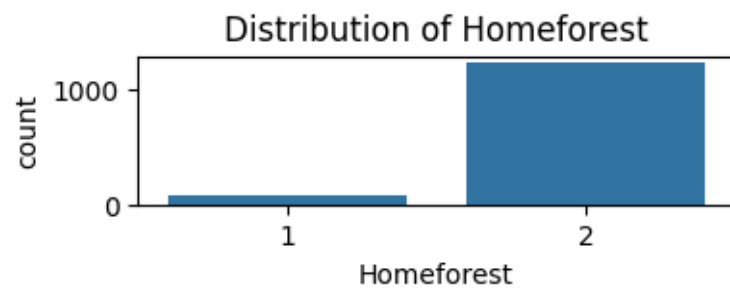
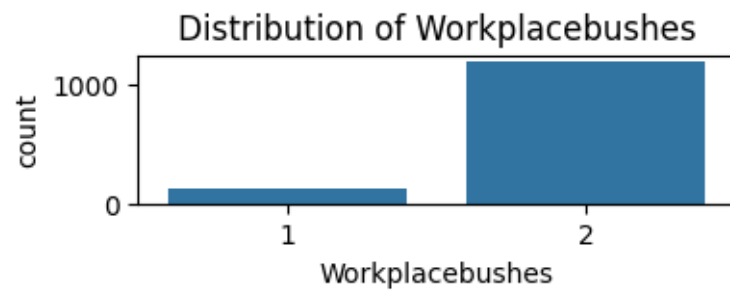
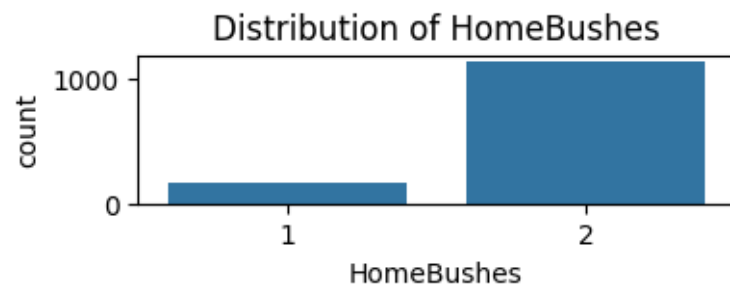
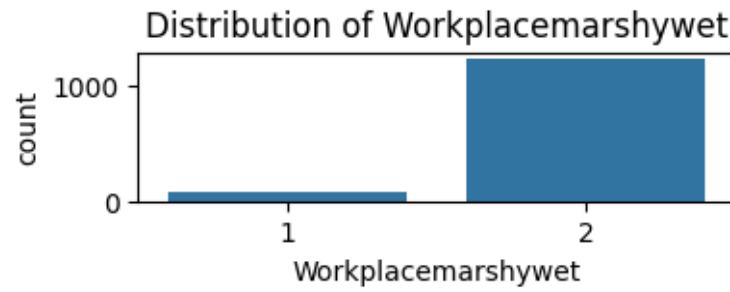


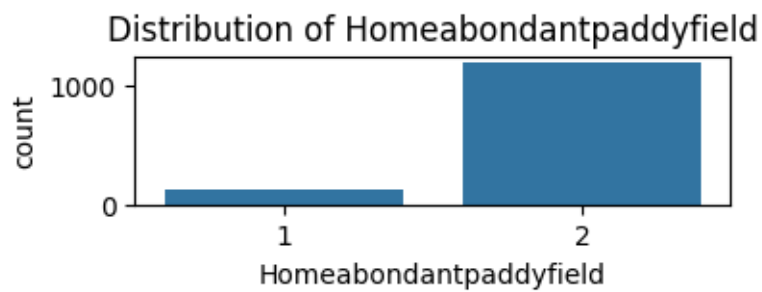
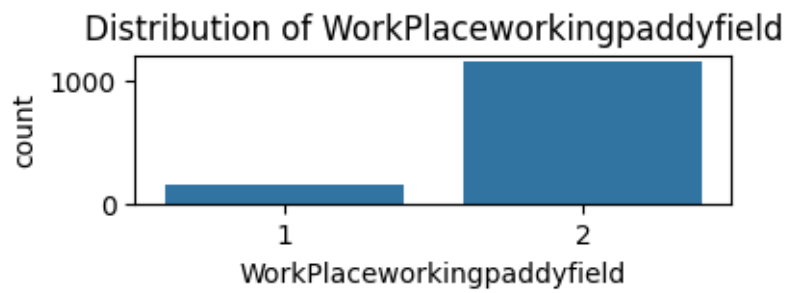
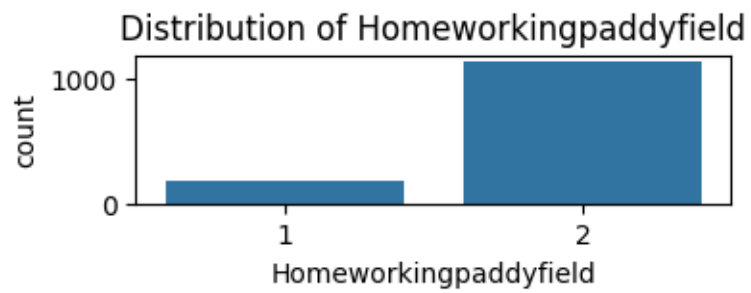
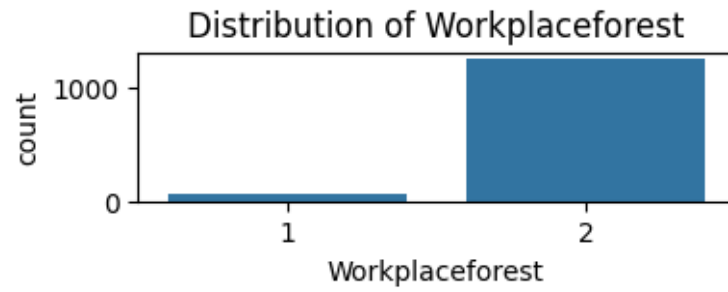
Distribution of WorkPlacepondlaketankoranyotherstagnantwatersource

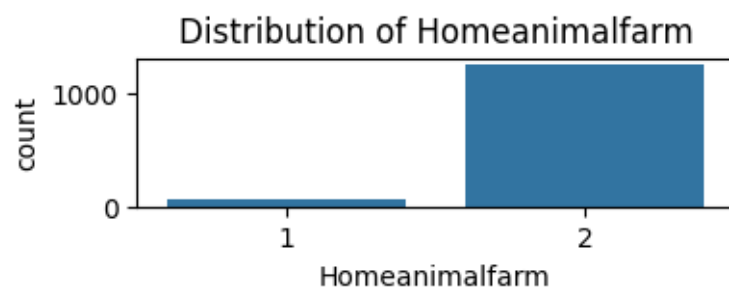
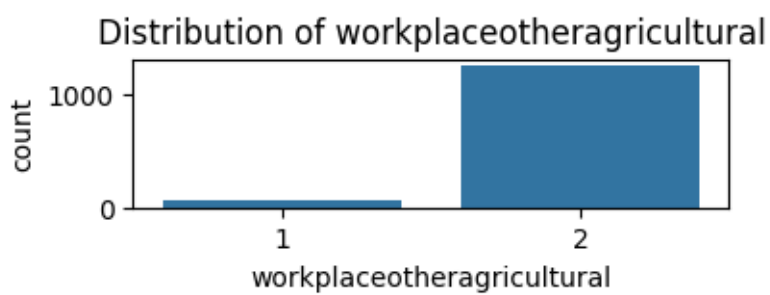
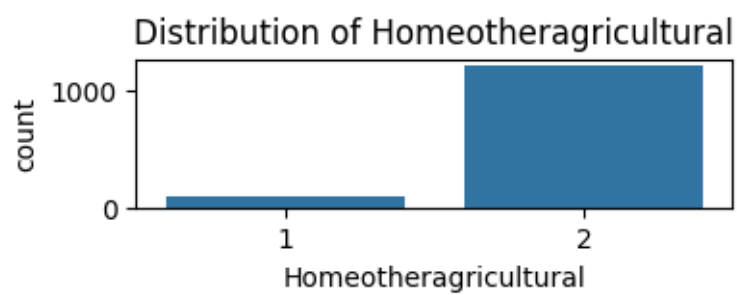
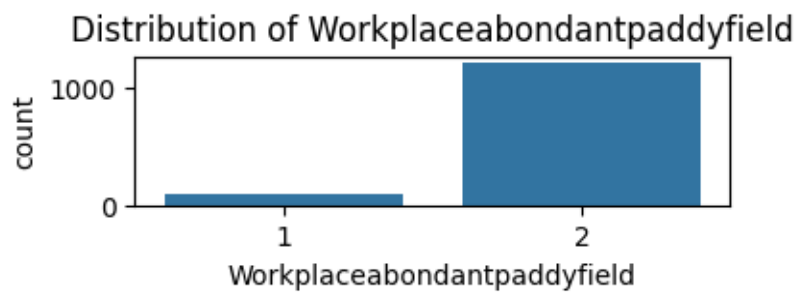


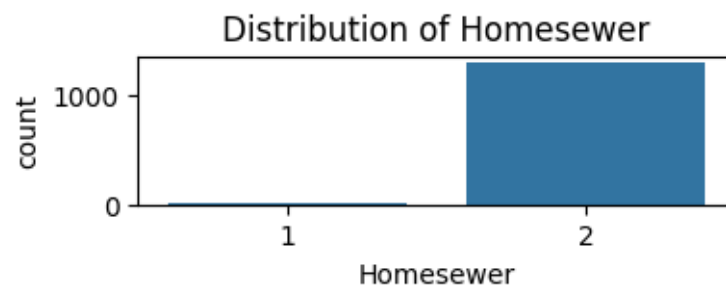
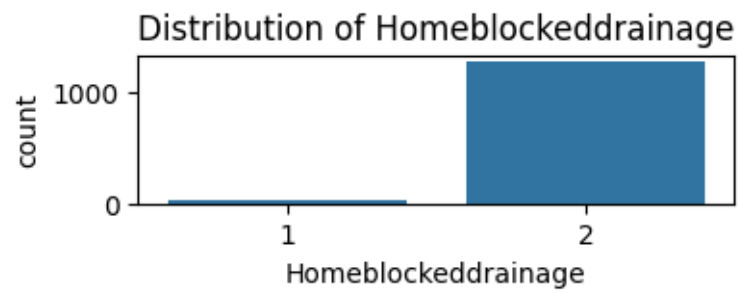
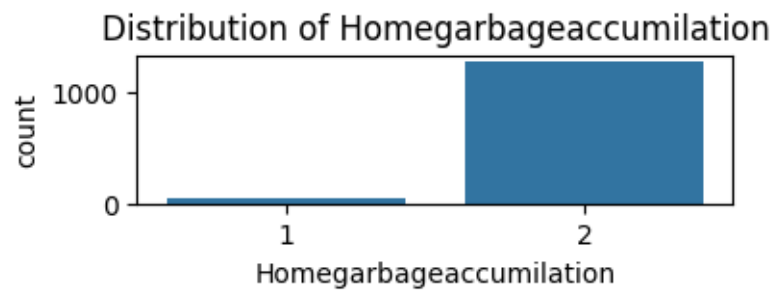
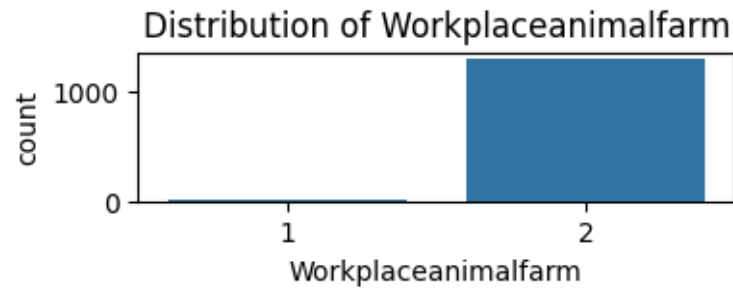
Distribution of Homemarshywetland

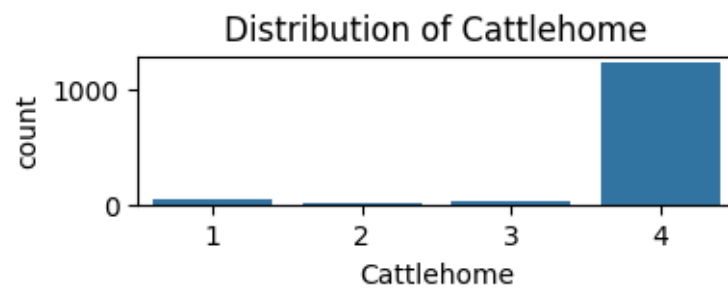
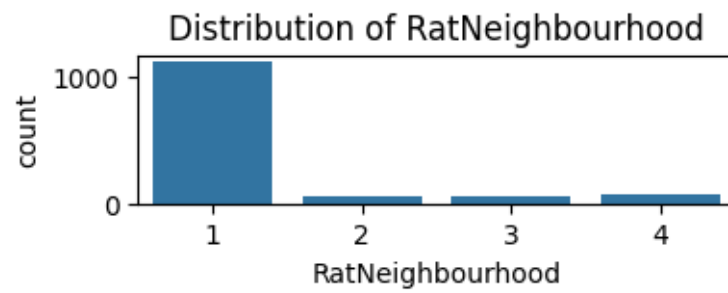
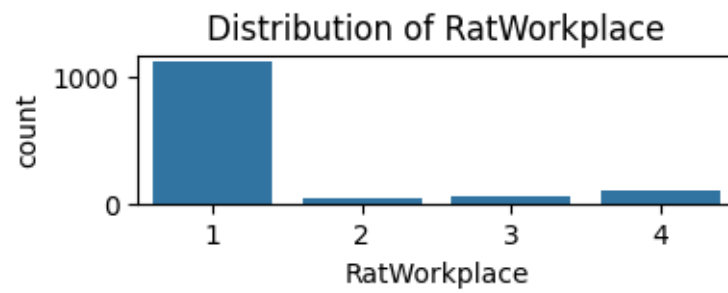
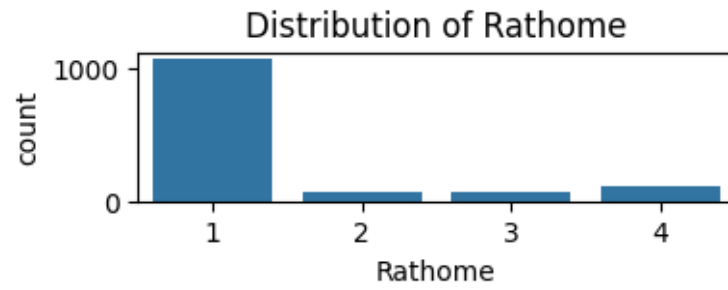


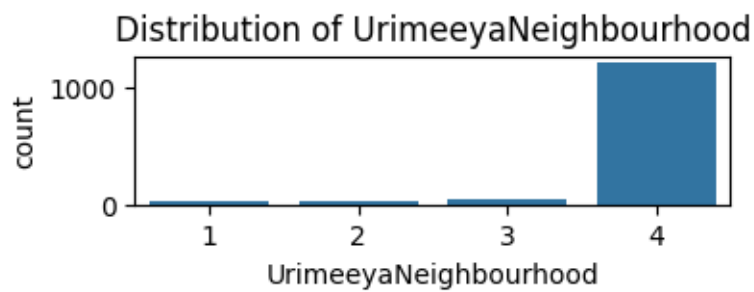
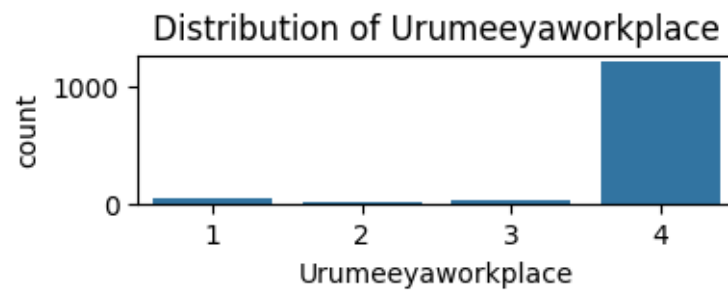
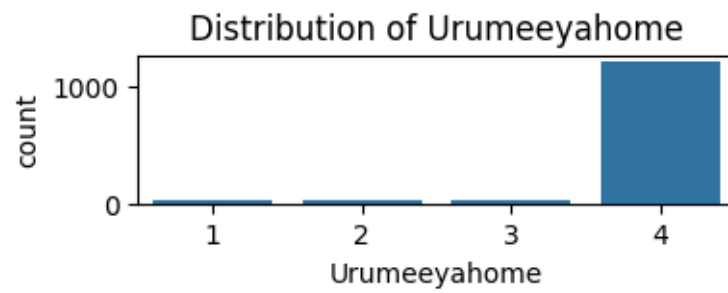
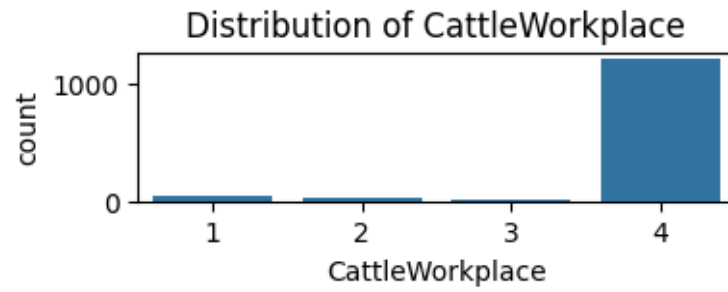


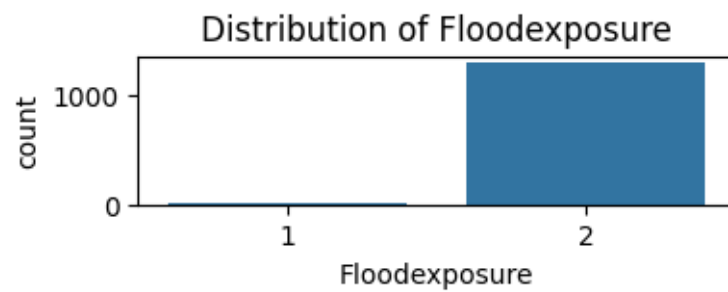
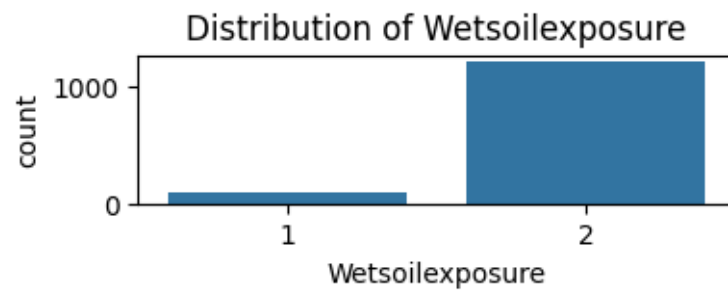
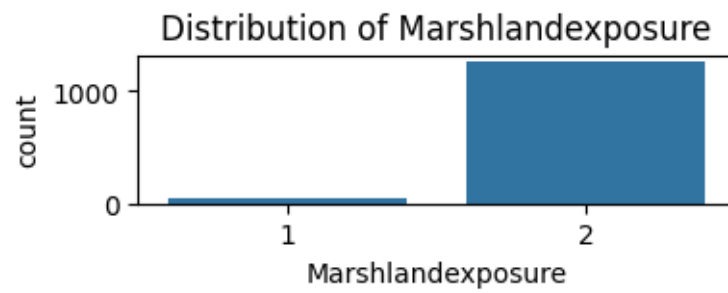
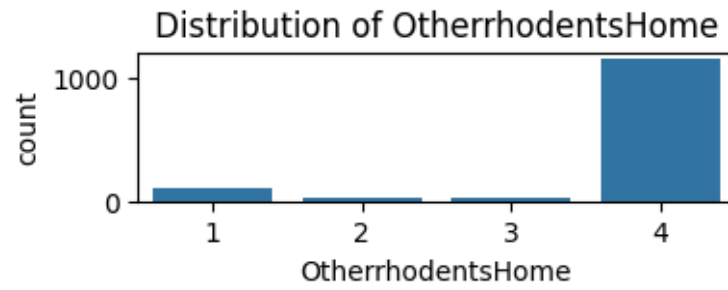


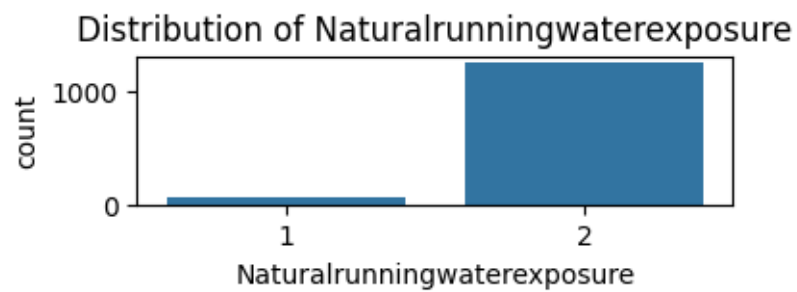
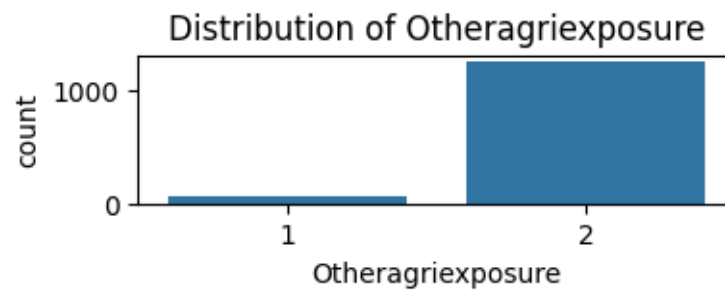
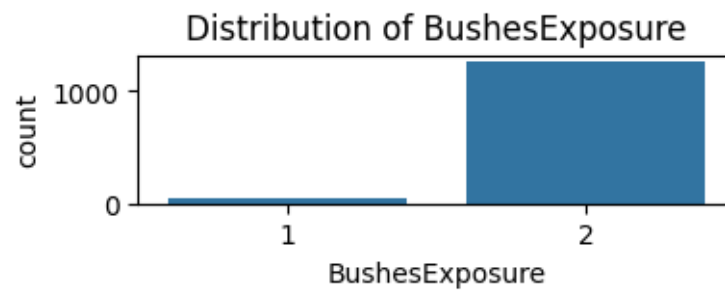
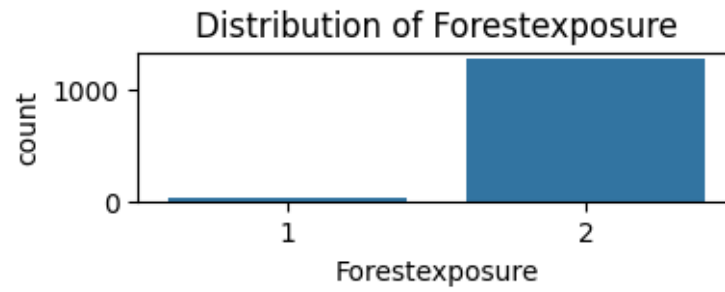


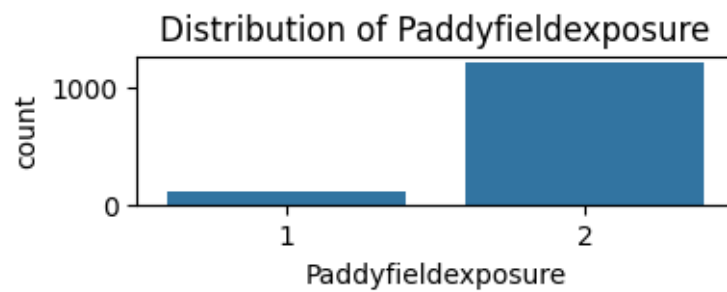
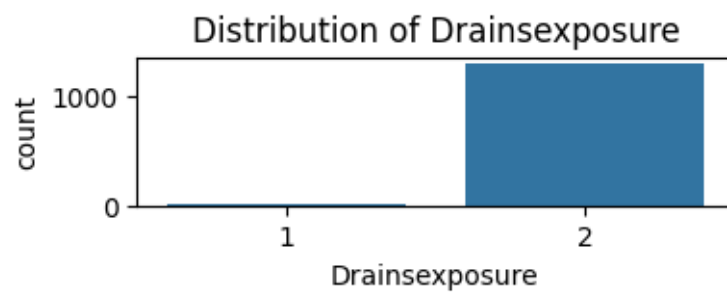
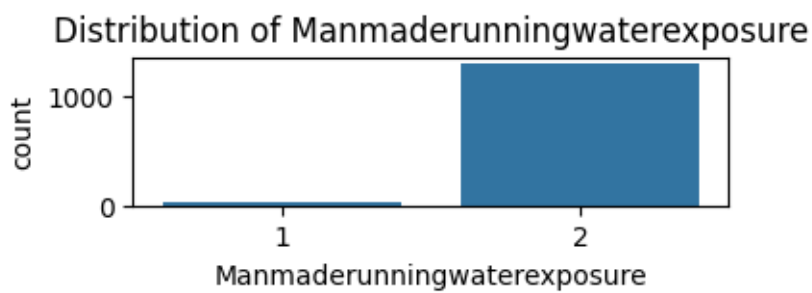
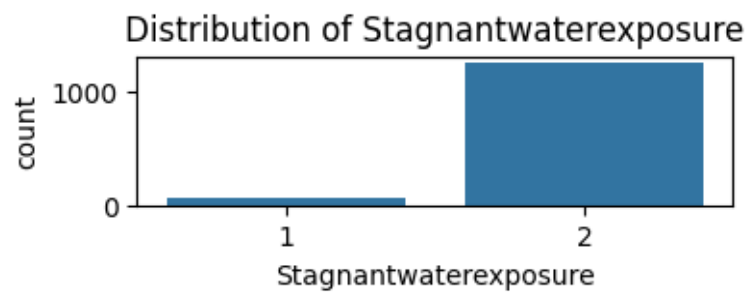


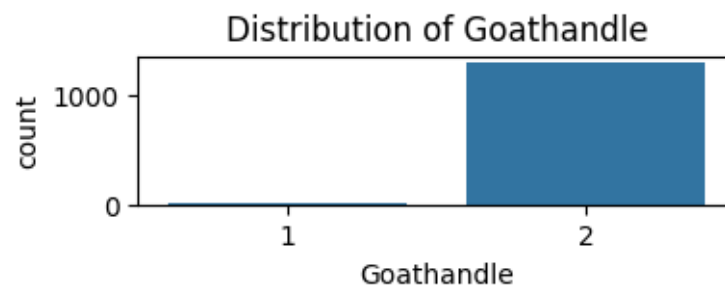
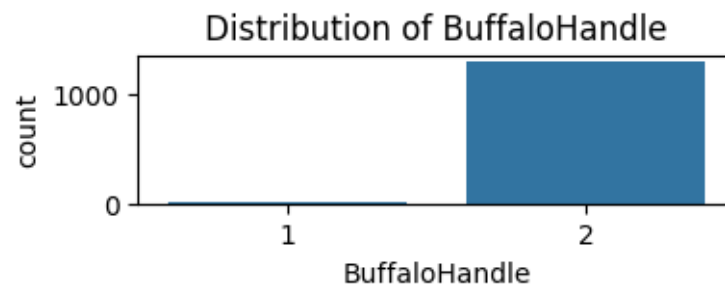
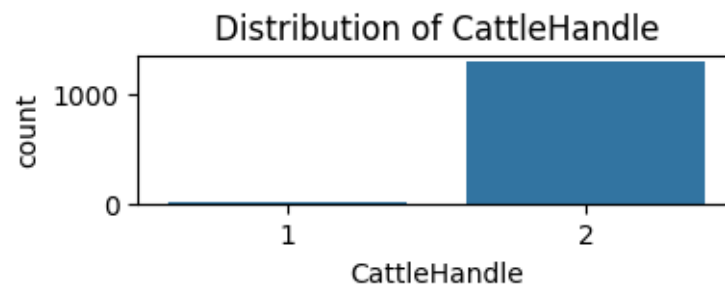
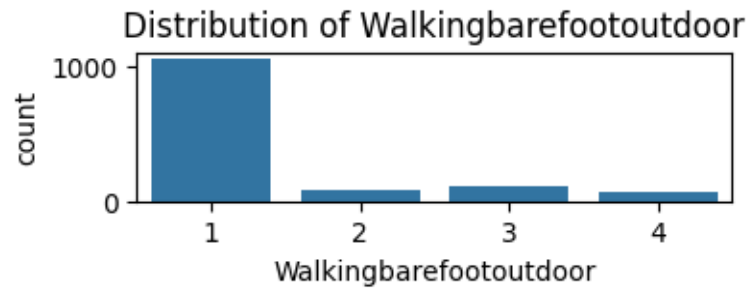


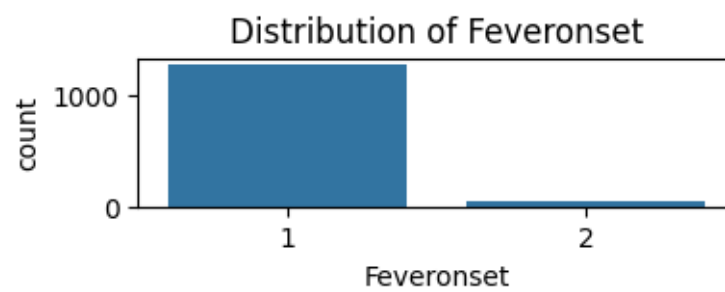
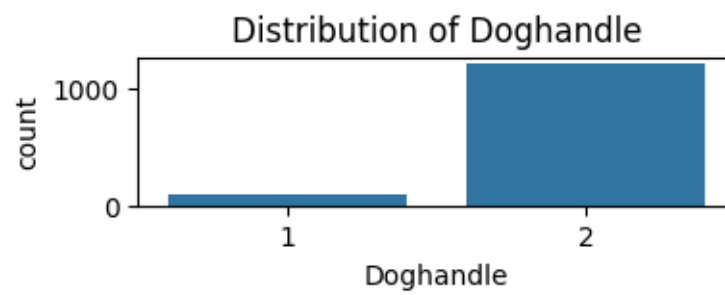
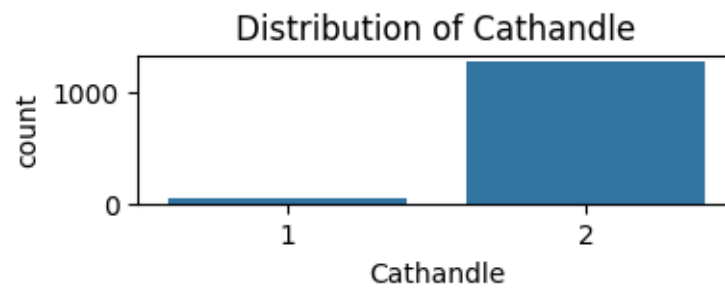
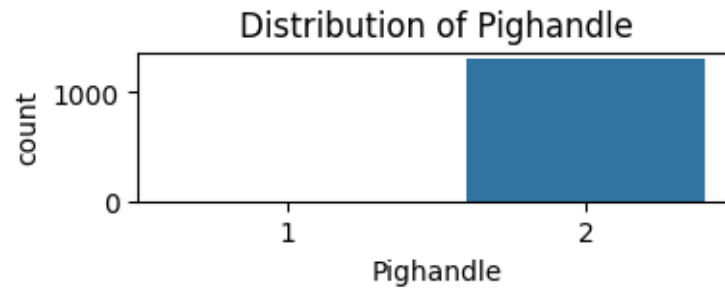


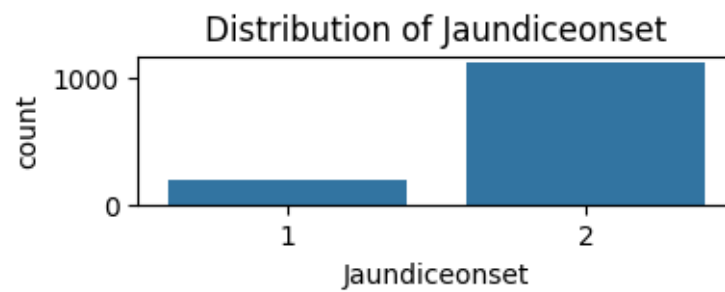
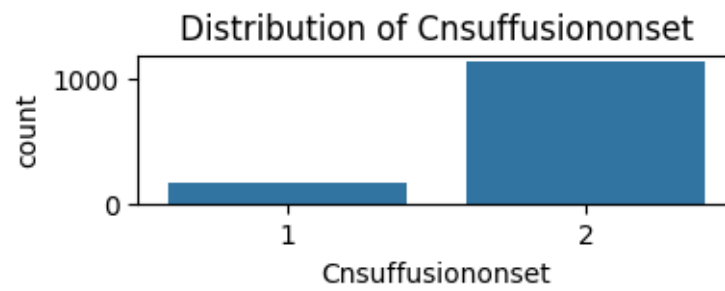
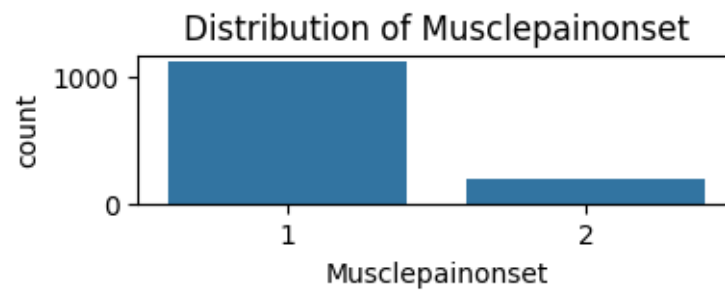
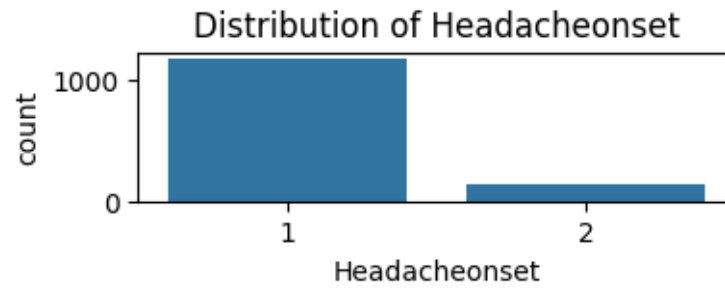


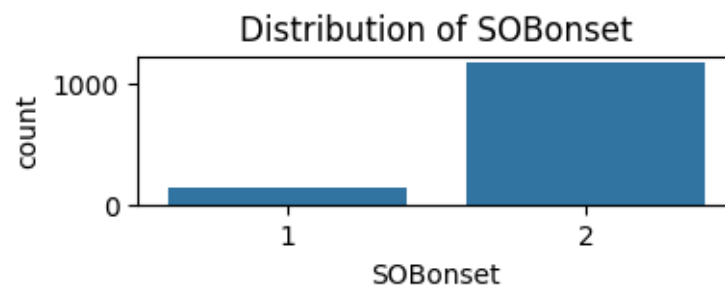
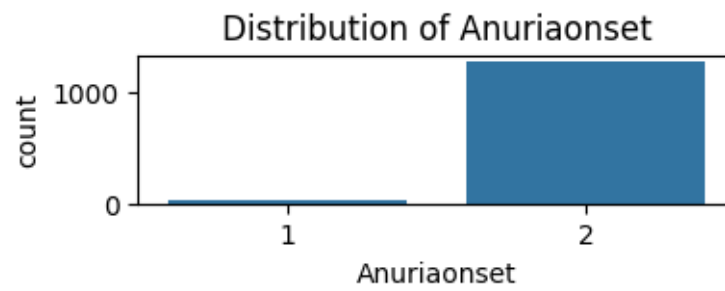
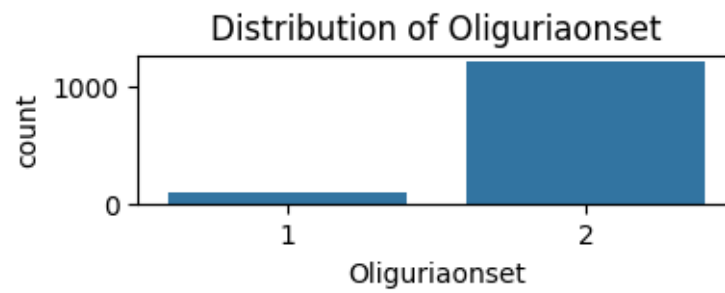
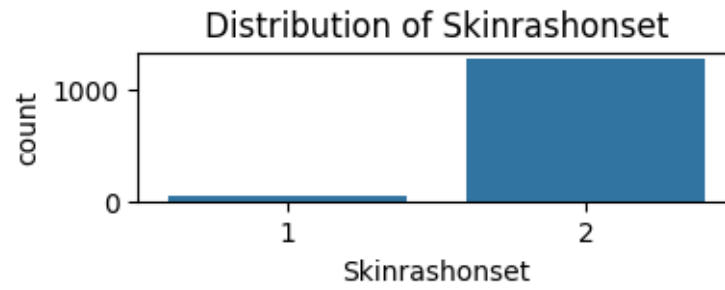


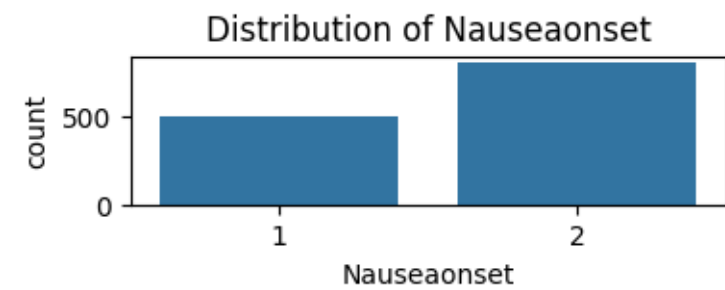
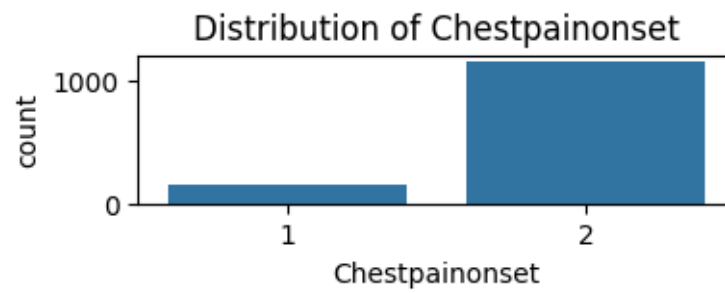
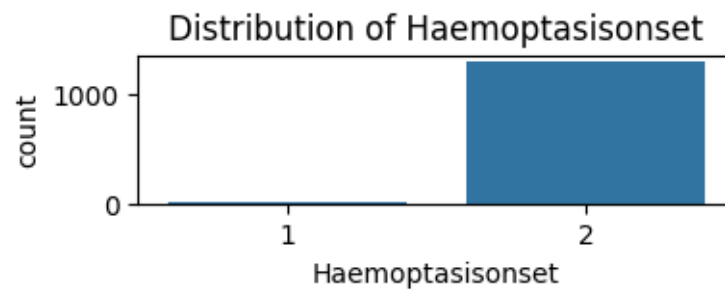
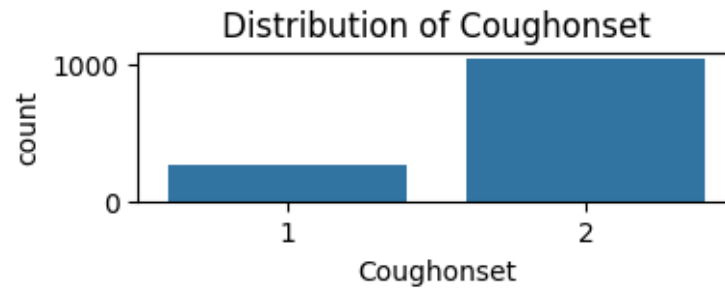


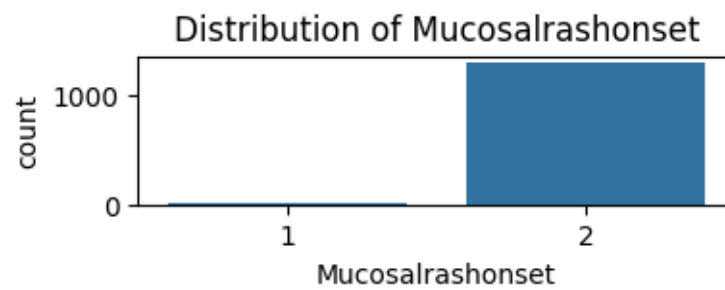
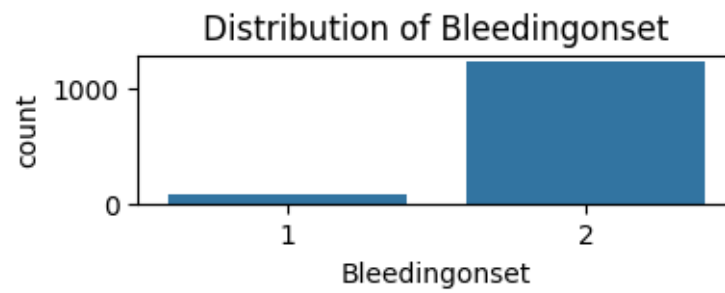
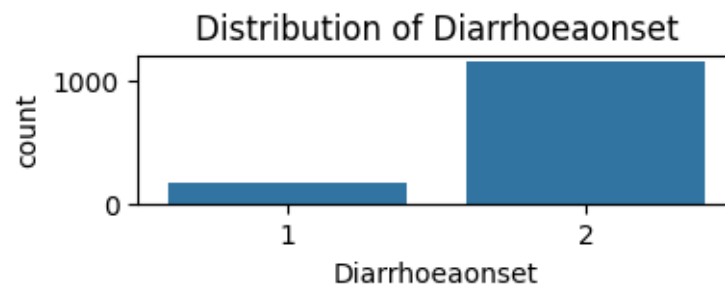
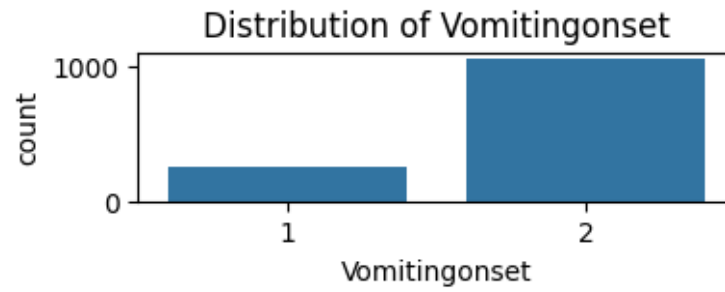


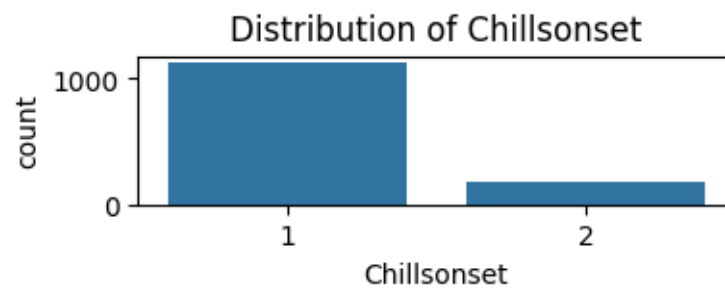
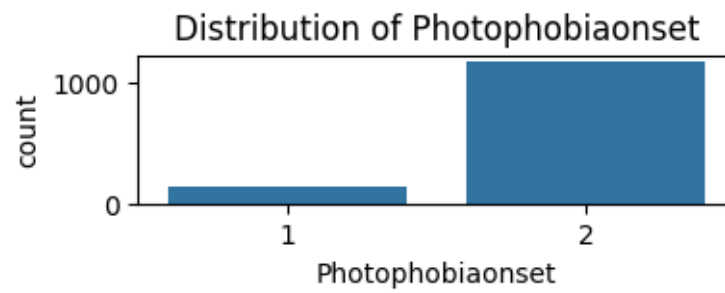
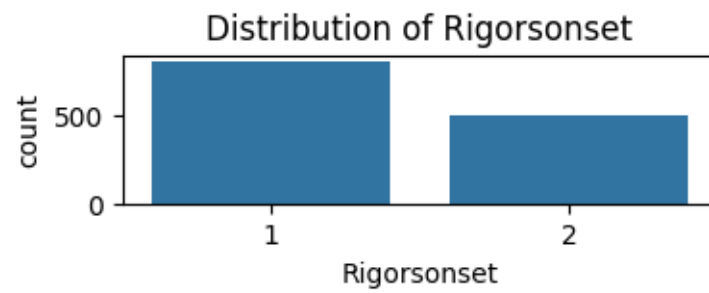
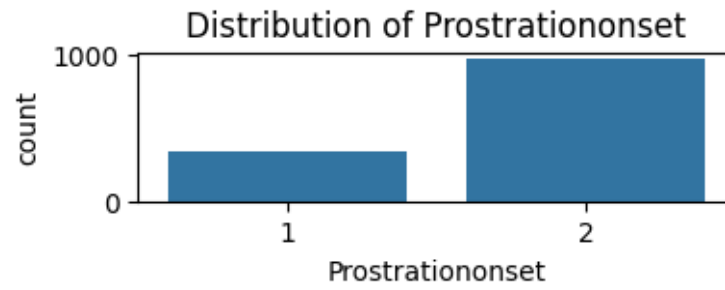


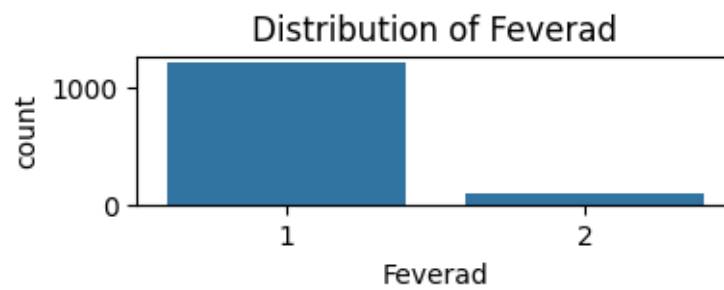
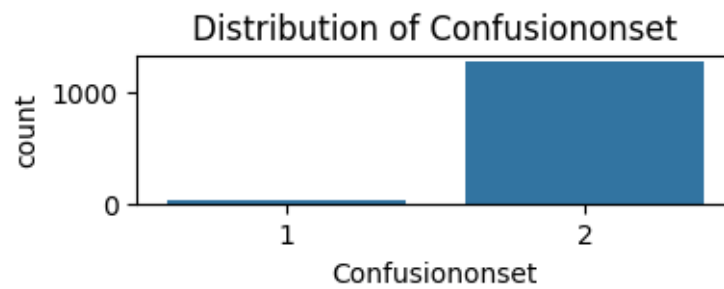
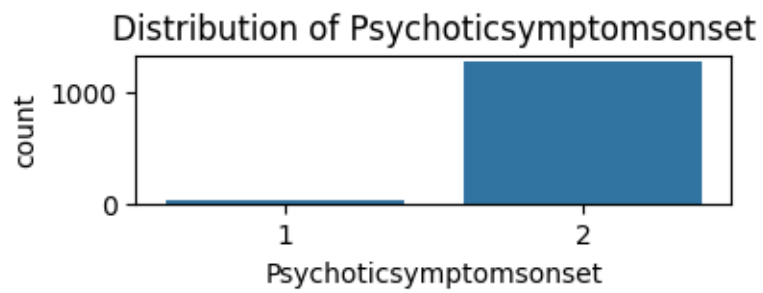
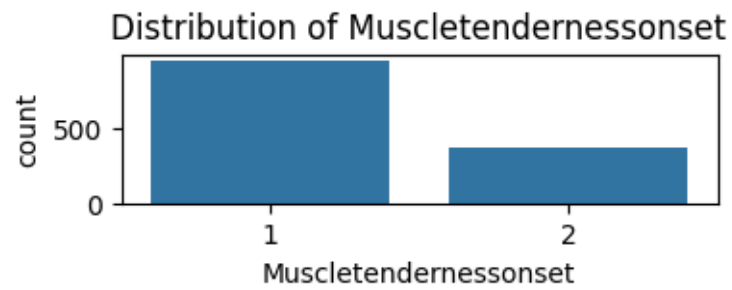


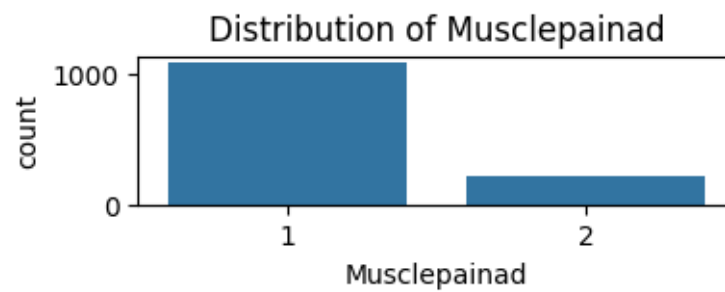
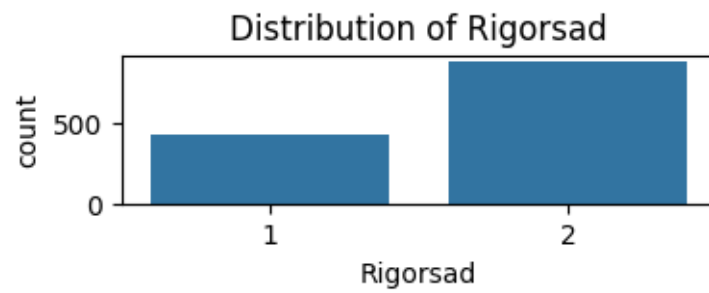
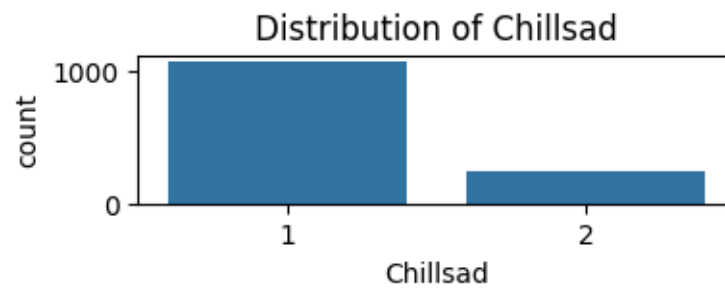
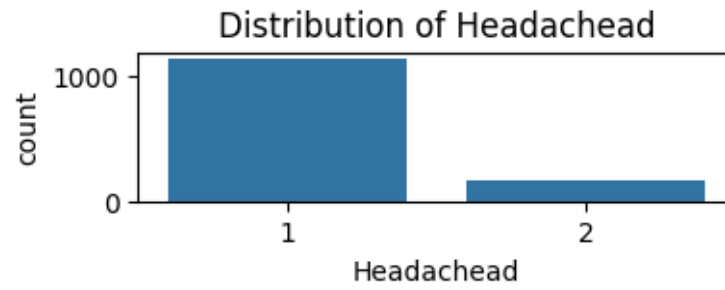


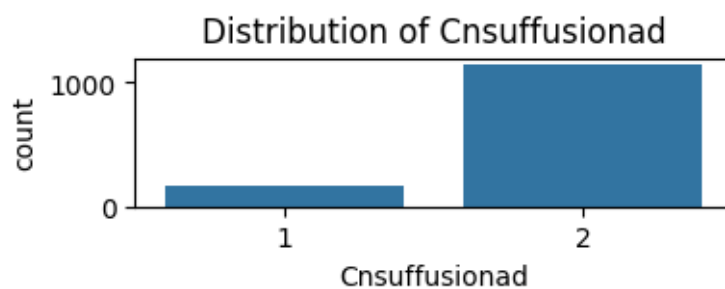
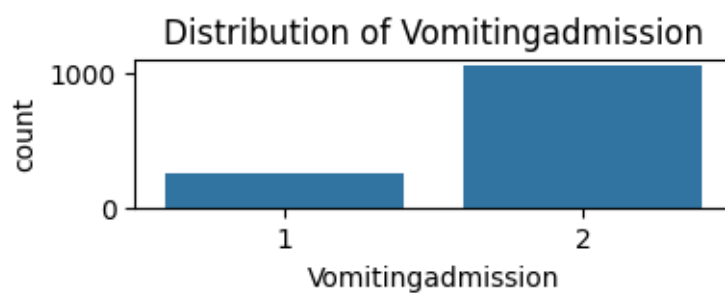
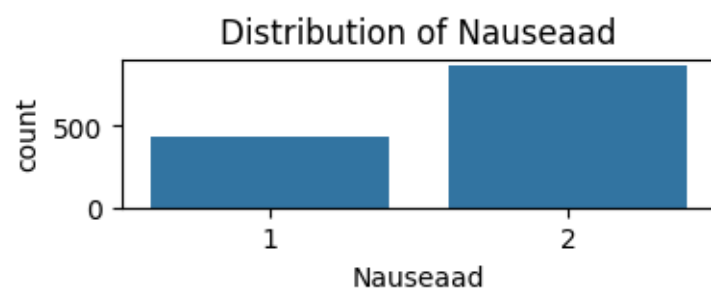
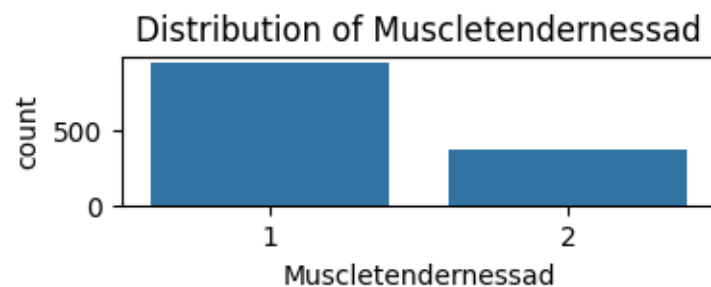


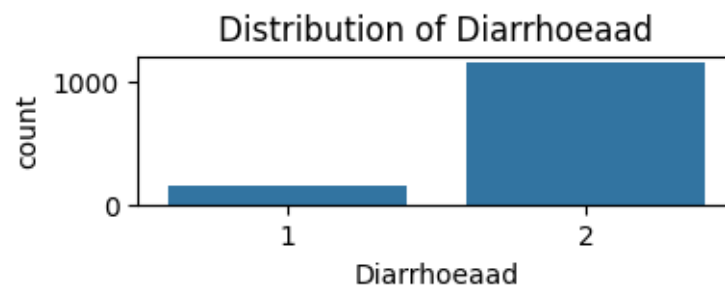
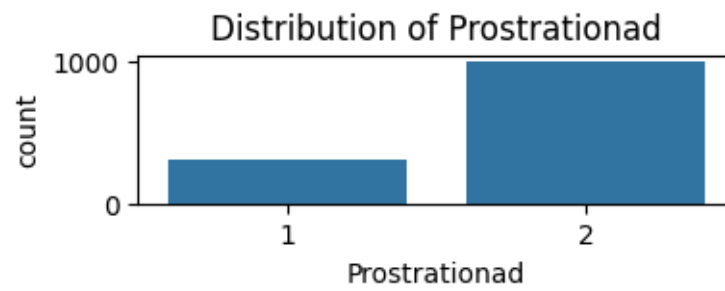
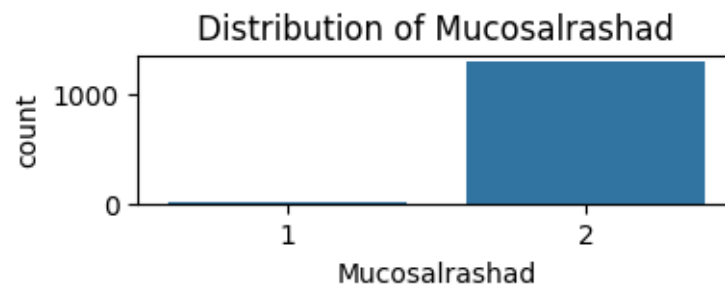
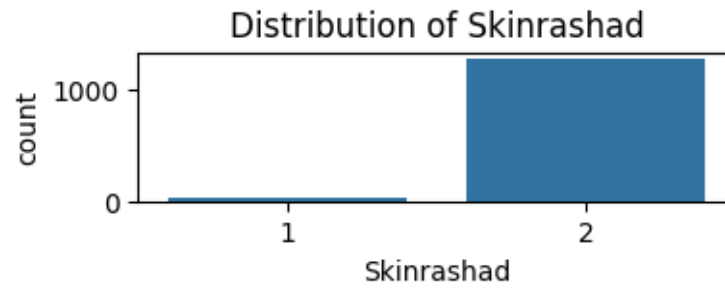


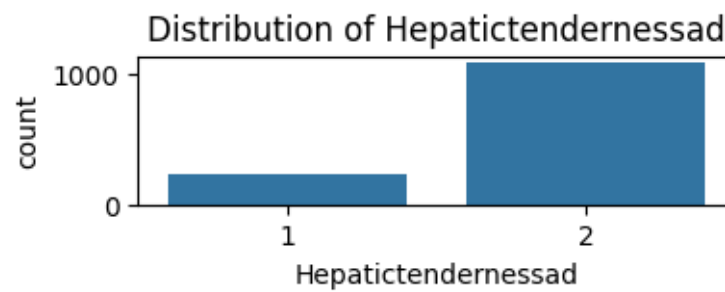
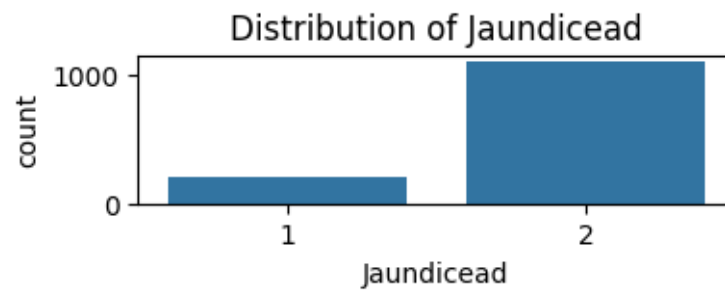
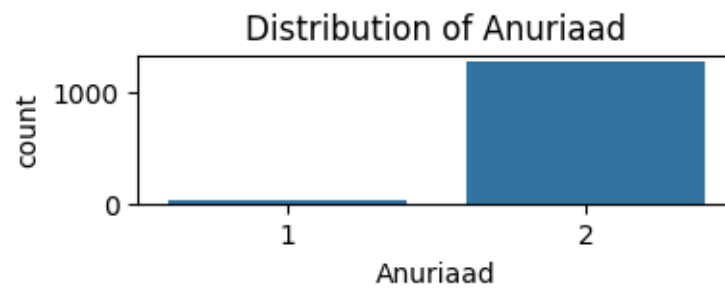
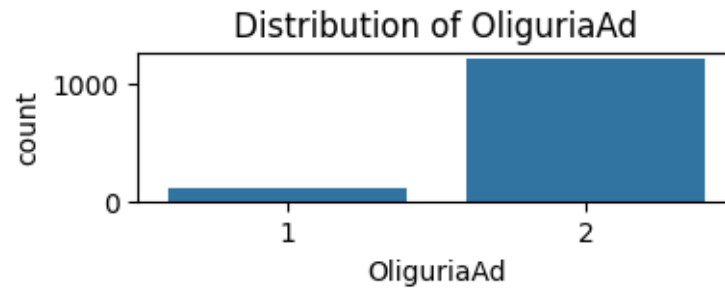


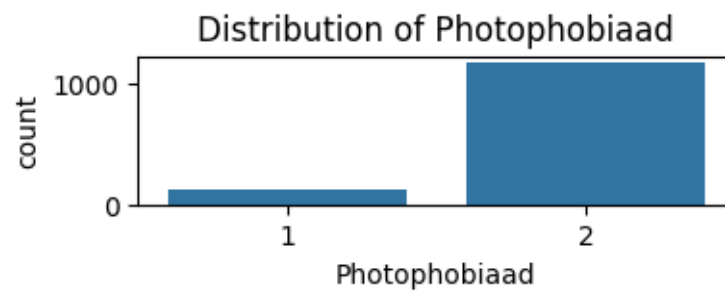
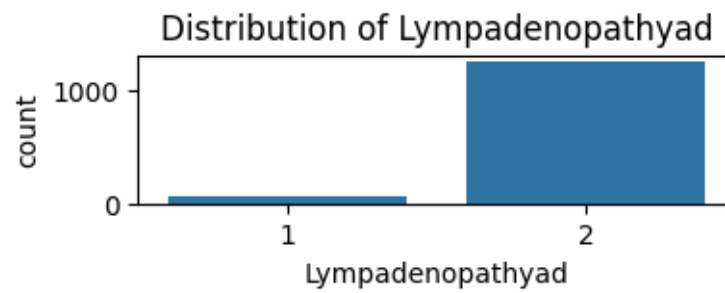
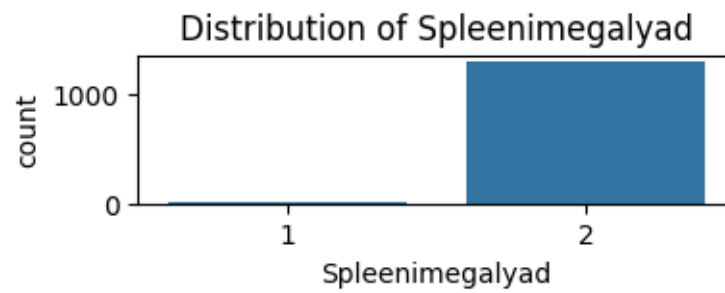
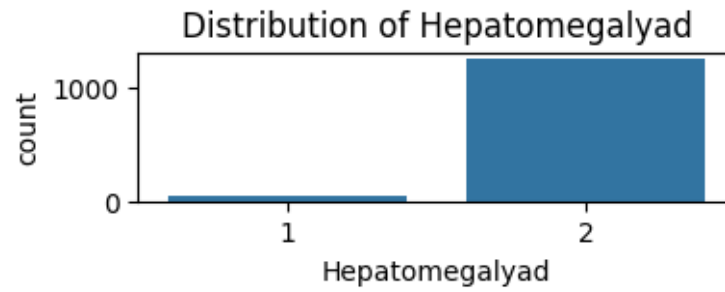


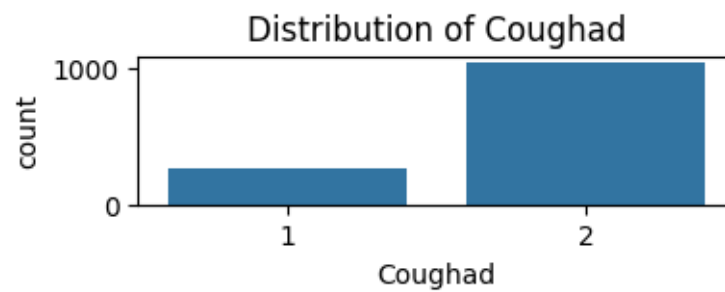
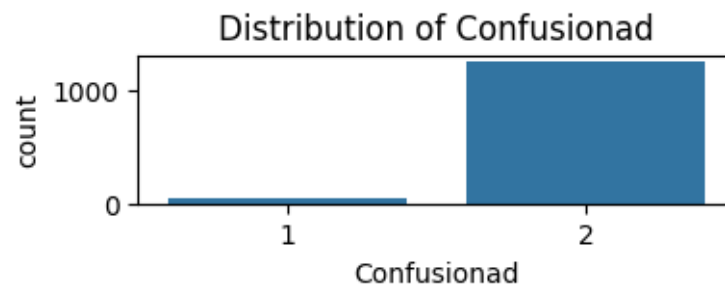
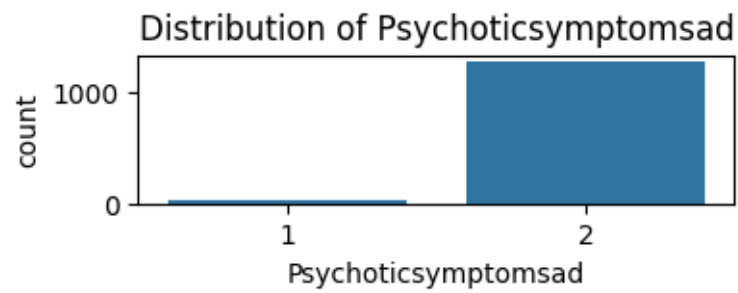
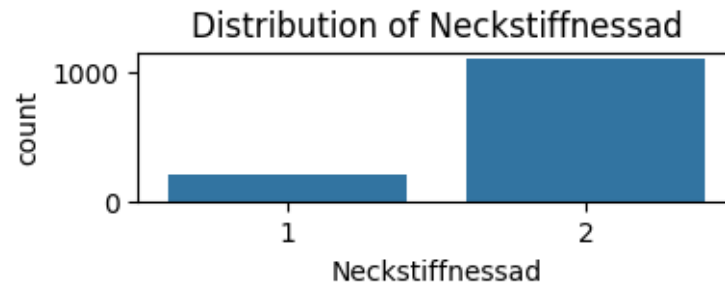


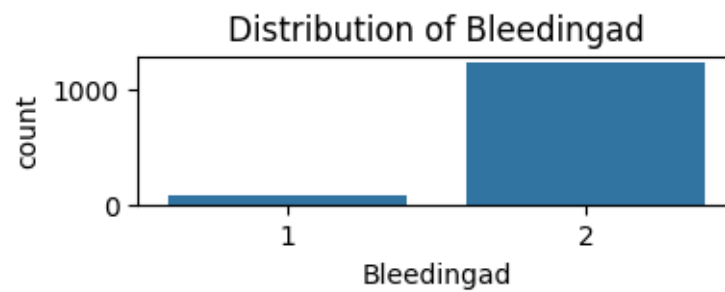
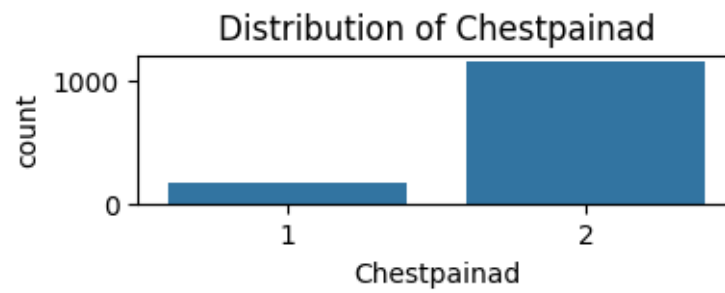
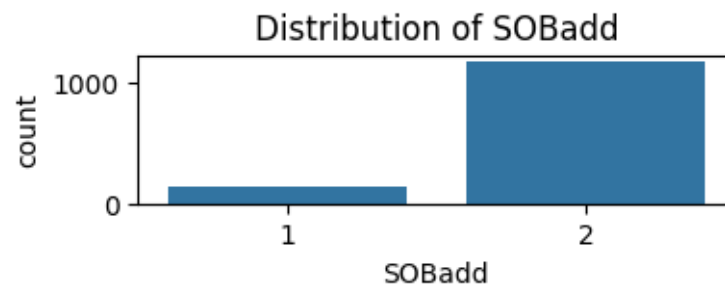
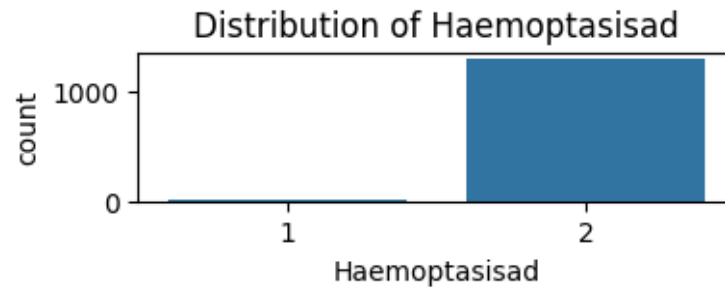


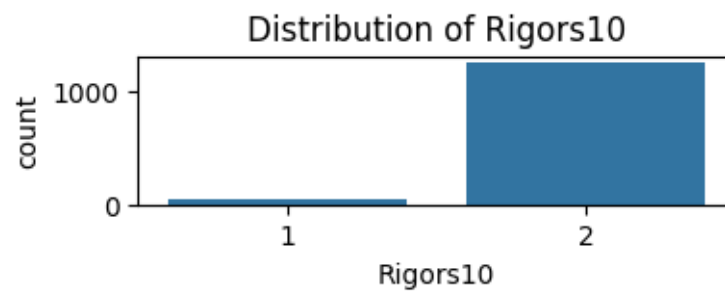
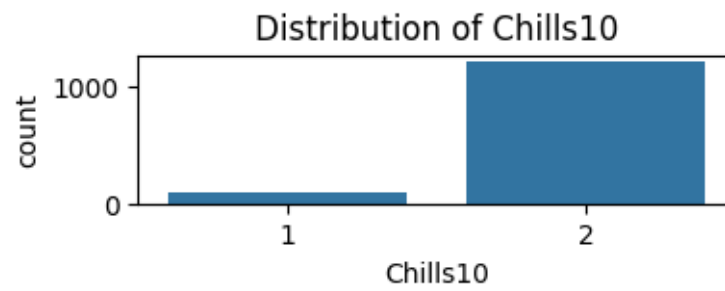
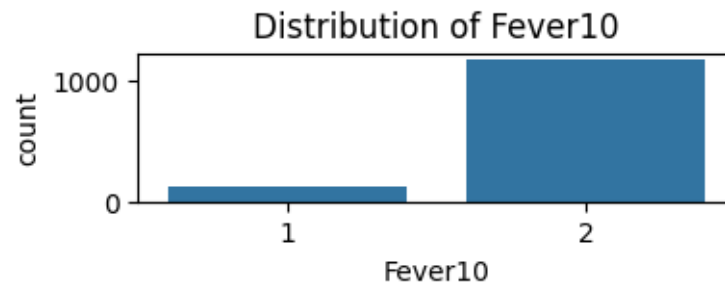
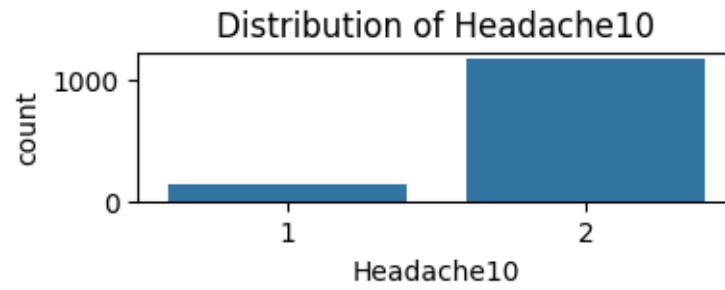


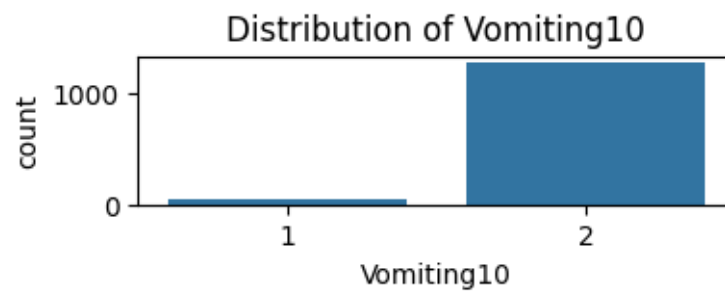
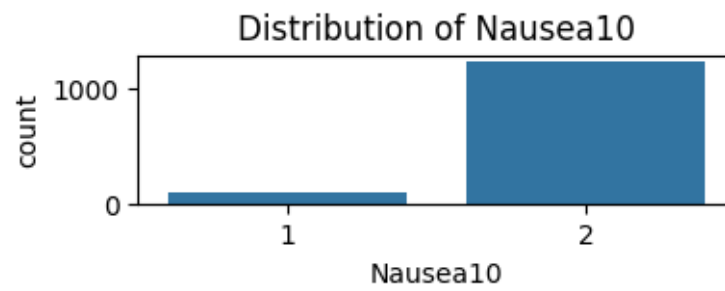
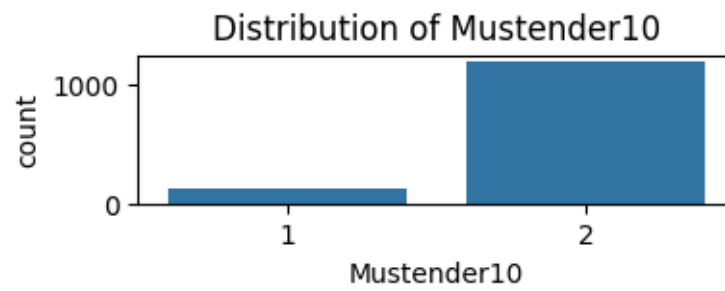
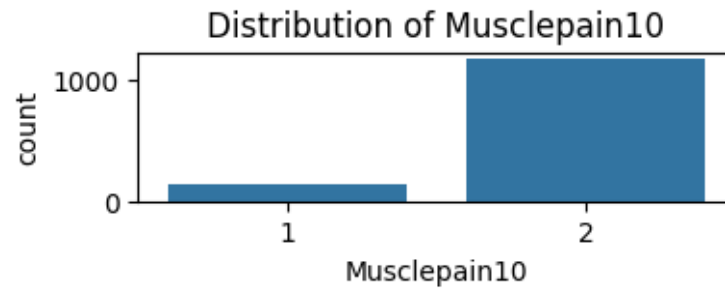


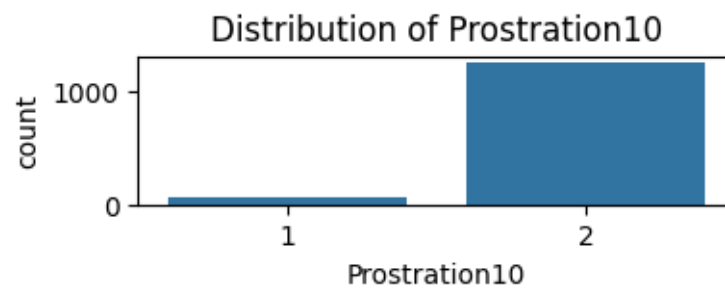
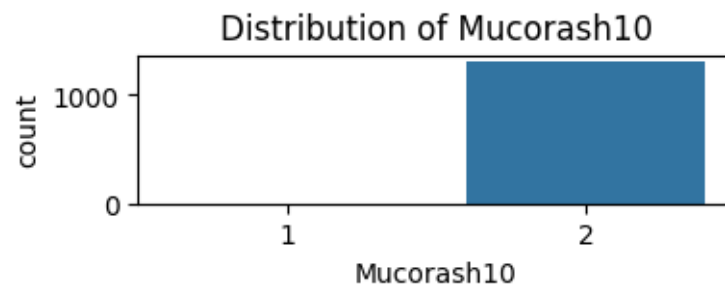
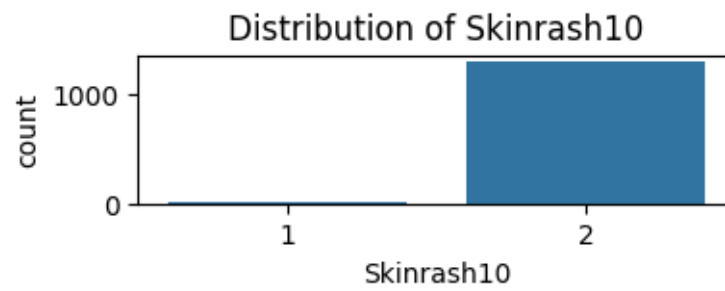
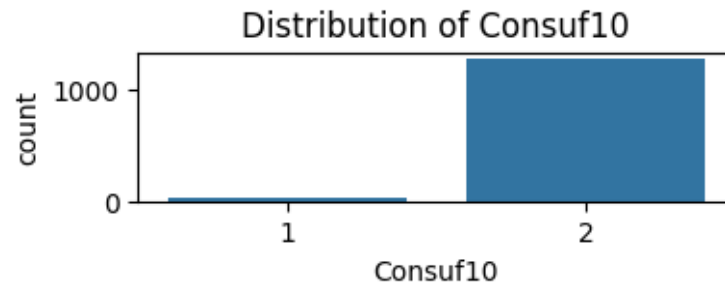


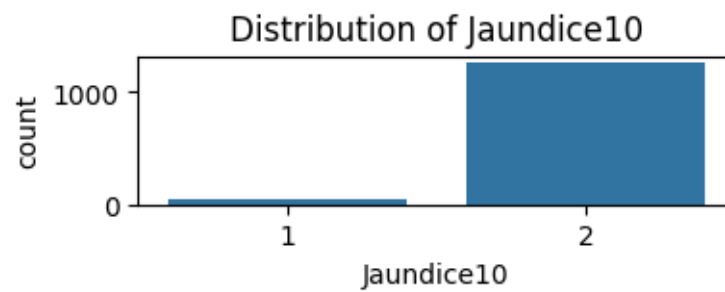
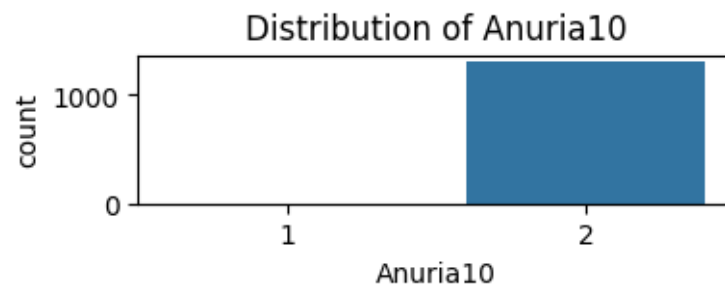
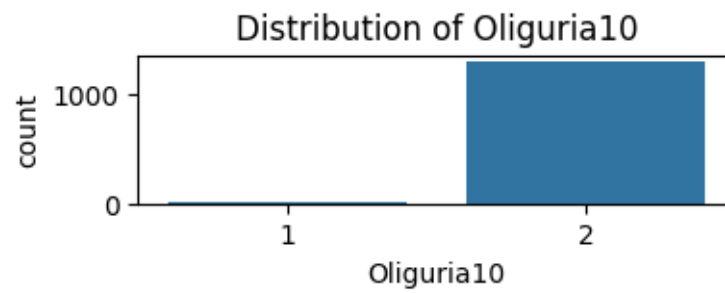
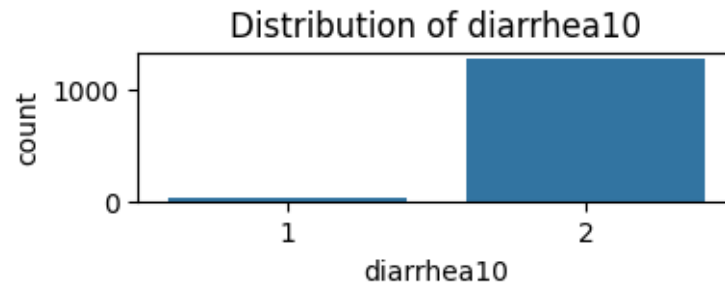


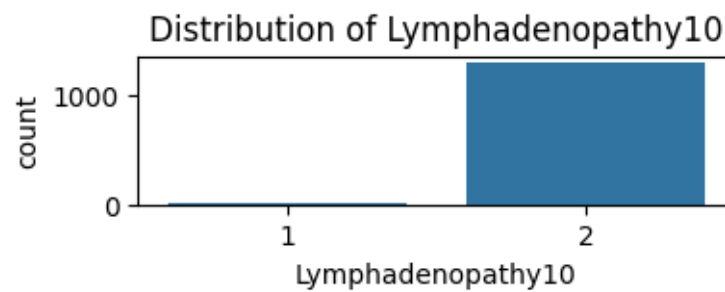
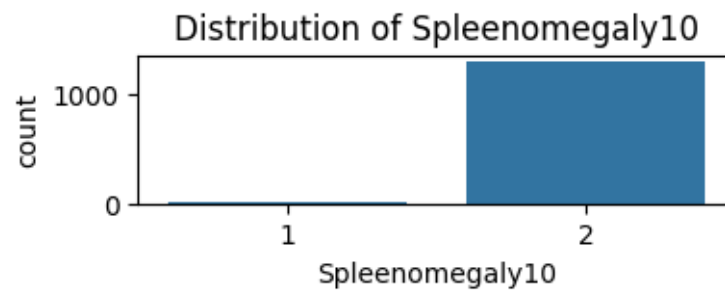
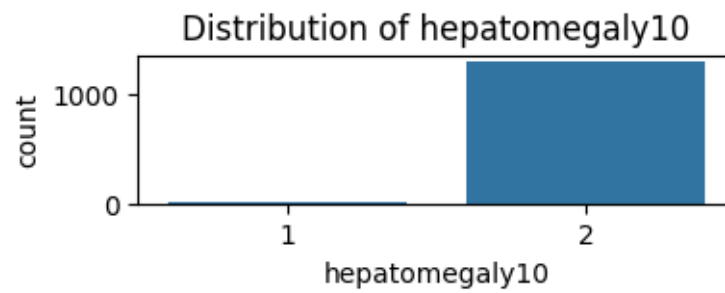
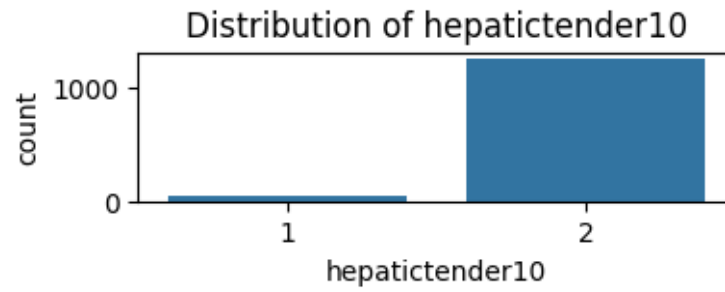


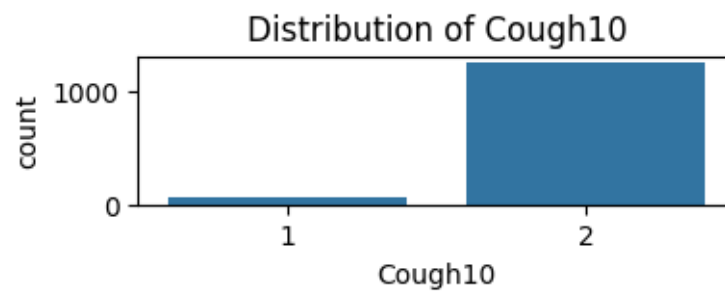
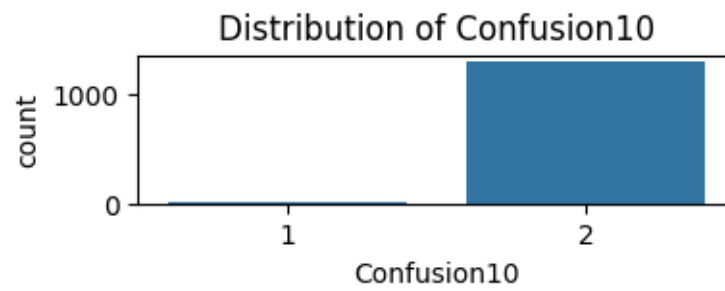
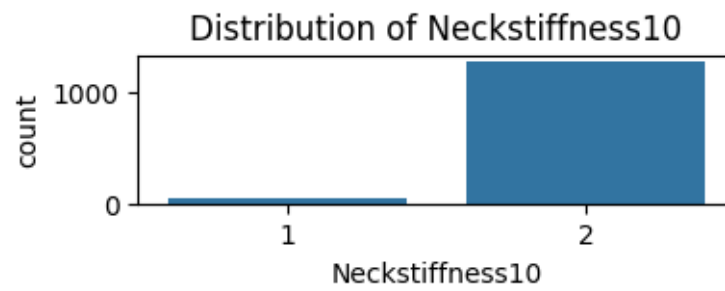
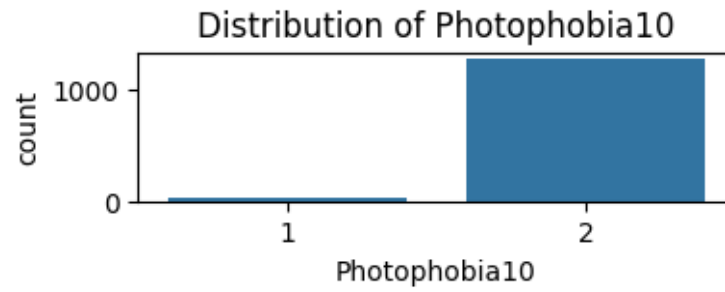


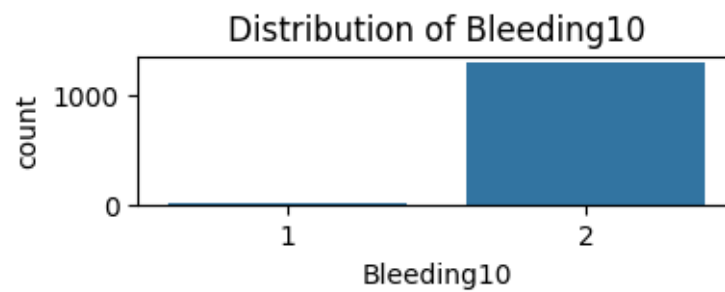
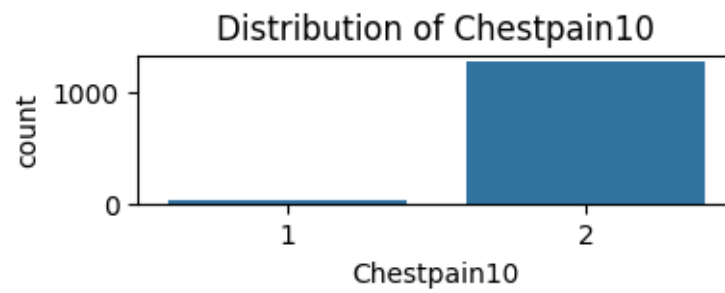
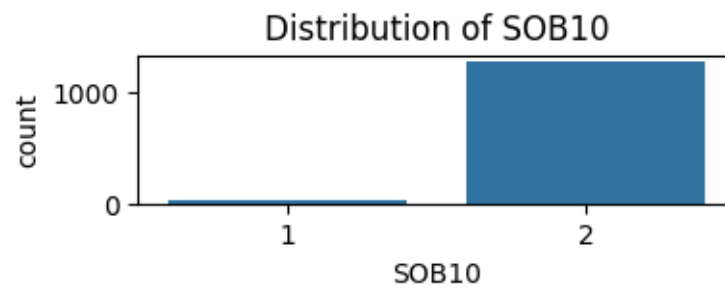
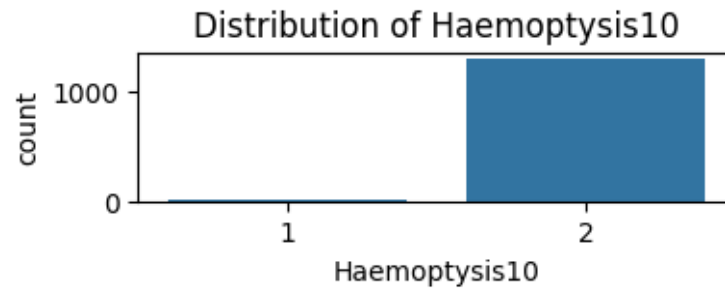


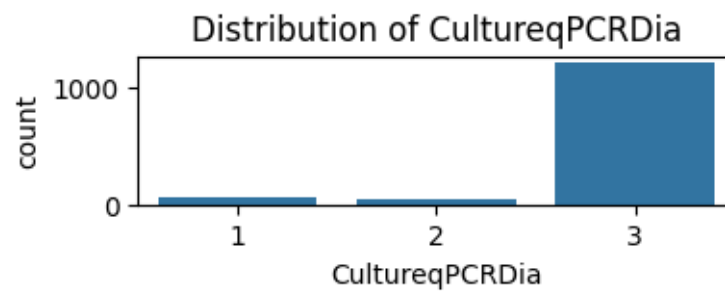
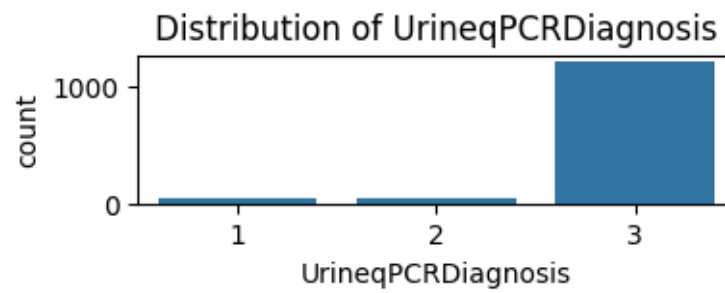
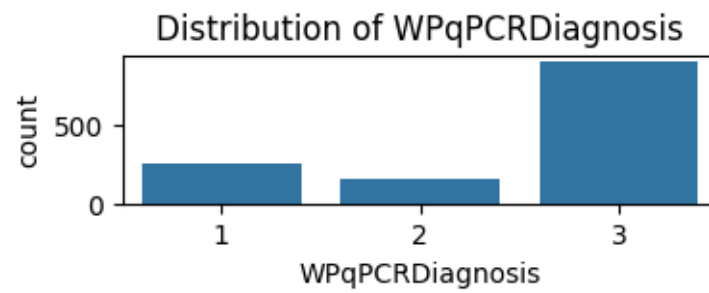
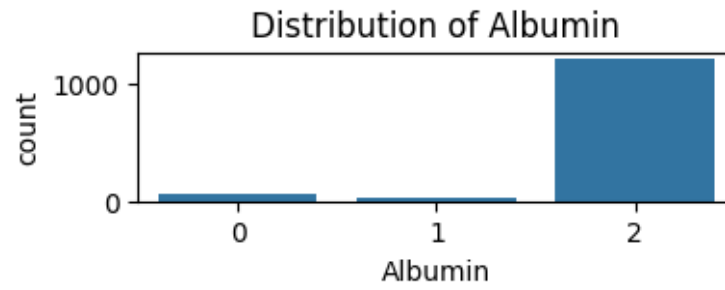


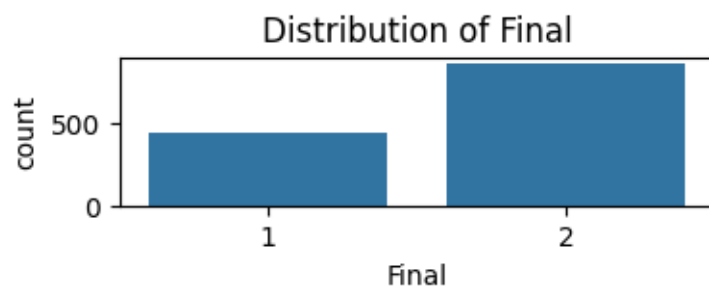
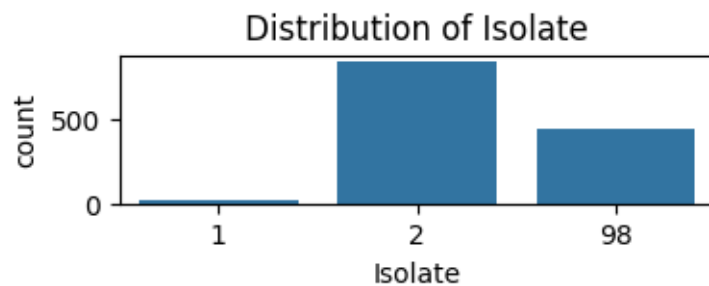
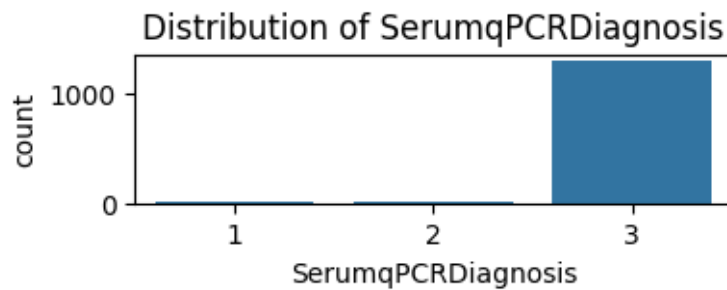












1.15 Perform SMOTE on 'Final' variable

```
[25]: from imblearn.over_sampling import SMOTE
      from collections import Counter

      #Split train data into X and y
      X_train = train_data.drop('Final', axis = 1)
      y_train = train_data['Final']

      smote = SMOTE(random_state = 42)
      X_train, y_train = smote.fit_resample(X_train, y_train)
```

```
print("SMOTE class distribution:", Counter(y_train))
```

SMOTE class distribution: Counter({2: 861, 1: 861})

1.16 Train Random Forest Classifier model

```
[26]: from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random_state = 42)
rf.fit(X_train, y_train)
```

[26]: RandomForestClassifier(random_state=42)

1.17 Perform hyperparameter tuning with cross-validation

```
[27]: from sklearn.model_selection import GridSearchCV

#Define hyperparameters
param_grid = {
    'n_estimators': [173, 174, 175],
    'max_depth': [22, 23, 24],
    'min_samples_split': [3, 4, 5],
    'min_samples_leaf': [1, 2]
}

#Perform GridSearchCV
grid_search = GridSearchCV(rf, param_grid, cv = 10, n_jobs = -1, scoring = 'accuracy')
grid_search.fit(X_train, y_train)

best_score = grid_search.best_score_
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

print("Best Score:", best_score)
print("Best Parameters:", best_params)
```

Best Score: 0.9251713940045704

Best Parameters: {'max_depth': 23, 'min_samples_leaf': 1, 'min_samples_split': 4, 'n_estimators': 174}

1.18 Predict model on train data

```
[28]: y_pred_train = best_model.predict(X_train)
```

1.19 Check accuracy for train data predictions

```
[29]: from sklearn.metrics import accuracy_score, confusion_matrix, \n      ↪ classification_report\n\naccuracy_train = accuracy_score(y_train, y_pred_train)\nprint("Accuracy: ", accuracy_train)\n\nconf_matrix_train = confusion_matrix(y_train, y_pred_train)\nprint("Confusion Matrix: \n", conf_matrix_train)\n\nclass_report_train = classification_report(y_train, y_pred_train)\nprint("Classification Report: \n", class_report_train)
```

Accuracy: 0.9814169570267132

Confusion Matrix:

```
[[829  32]
```

```
[  0 861]]
```

Classification Report:

	precision	recall	f1-score	support
1	1.00	0.96	0.98	861
2	0.96	1.00	0.98	861
accuracy			0.98	1722
macro avg	0.98	0.98	0.98	1722
weighted avg	0.98	0.98	0.98	1722

1.20 Split the train dataset itself into train and validation sets

```
[30]: from sklearn.model_selection import train_test_split\n\nX_train_tr, X_test_tr, y_train_tr, y_test_tr = train_test_split(X_train, \n      ↪ y_train, test_size = 0.2, random_state = 42)
```

1.21 Perform SMOTE on train and validation sets again

```
[31]: X_train_tr, y_train_tr = smote.fit_resample(X_train_tr, y_train_tr)\n\nprint("Train-SMOTE class distribution", Counter(y_train_tr))
```

Train-SMOTE class distribution Counter({1: 697, 2: 697})

```
[32]: X_train_tr.shape
```

```
[32]: (1394, 172)
```


1.22 Train RF inside train dataset¶

```
[33]: rf.fit(X_train_tr, y_train_tr)
```

```
[33]: RandomForestClassifier(random_state=42)
```

1.23 Perform hyperparameter tuning with CV inside train dataset

```
[34]: grid_search_tr = GridSearchCV(rf, param_grid, cv = 10, n_jobs = -1, scoring = 'accuracy')
      grid_search_tr.fit(X_train_tr, y_train_tr)

      best_score_tr = grid_search_tr.best_score_
      best_params_tr = grid_search_tr.best_params_
      best_model_tr = grid_search_tr.best_estimator_

      print("Best Score:", best_score_tr)
      print("Best Parameters:", best_params_tr)
```

```
Best Score: 0.9239928057553957
```

```
Best Parameters: {'max_depth': 23, 'min_samples_leaf': 1, 'min_samples_split': 4, 'n_estimators': 174}
```

1.24 Predict model inside train dataset

```
[35]: y_pred_train_tr = best_model_tr.predict(X_train_tr)
```

1.25 Check accuracy for train data predictions inside train dataset

```
[36]: accuracy_train_tr = accuracy_score(y_train_tr, y_pred_train_tr)
      print("Accuracy: ", accuracy_train_tr)
      conf_matrix_train_tr = confusion_matrix(y_train_tr, y_pred_train_tr)
      print("Confusion Matrix: \n", conf_matrix_train_tr)
      class_report_train_tr = classification_report(y_train_tr, y_pred_train_tr)
      print("Classification Report: \n", class_report_train_tr)
```

```
Accuracy: 0.9827833572453372
```

```
Confusion Matrix:
```

```
[[673  24]
```

```
 [  0 697]]
```

```
Classification Report:
```

	precision	recall	f1-score	support
1	1.00	0.97	0.98	697
2	0.97	1.00	0.98	697
accuracy			0.98	1394

macro avg	0.98	0.98	0.98	1394
weighted avg	0.98	0.98	0.98	1394

1.26 Load test dataset

```
[37]: test_data_whole = pd.read_csv('test.csv').drop('ID', axis = 1)

#Get train dataset categorical columns without 'Final' variable
categorical_vars_filtered = [col for col in categorical_vars if col != 'Final']

#Get columns for test data that are included in train data after preprocessing
test_data = test_data_whole[categorical_vars_filtered + numerical_vars]
print(test_data.shape)
```

(347, 172)

1.27 Preprocess test data into train data format

```
[38]: #Handle duplicates
test_data = test_data.drop_duplicates()

[39]: #Replace values starting with 'fiel' or 'occ' with 99
test_data = test_data.map(lambda x: 99 if isinstance(x, str) and (x.lower().
↪startswith('fiel') or x.lower().startswith('occ')) else x)
print(test_data)
```

	Hospital	Sample	ICU	OPD	Sex	Prophylactics	Pasttreatments	\
0	1	1	2	2	1	2	2	
1	1	1	2	2	1	2	1	
2	1	1	2	2	1	1	1	
3	1	1	2	2	2	2	1	
4	1	1	2	1	1	99	99	
..	
342	8	1	2	2	1	2	1	
343	8	1	2	2	1	2	2	
344	8	1	2	2	1	2	2	
345	8	1	2	2	1	2	2	
346	8	1	1	2	2	2	2	

	Pastantibiotics	Chronicillness	Possibleexposure	...	ESR	Redcells	\
0	99		2	1	99.0	99	
1	3		2	2	99.0	99	
2	1		2	1	99.0	99	
3	1		2	1	99.0	99	
4	99		99	99	99.0	99	
..	
342	1		2	1	99.0	99	

343	2	2	1 ...	99.0	99
344	2	2	1 ...	99.0	99
345	2	2	1 ...	99.0	12
346	2	2	1 ...	99.0	99

	Na	K	AST	T.Bilirubin	S.creatinine	B.urea	ALP	Puscells
0	99	99.00	99.0	99.0	99.0	99.0	99	99
1	99	99.00	177.0	99.0	88.4	8.0	99	99
2	99	99.00	99.0	99.0	99.0	99.0	99	99
3	99	99.00	99.0	99.0	99.0	99.0	99	99
4	99	99.00	99.0	99.0	99.0	99.0	99	99
..
342	143	3.40	23.7	8.2	65.0	12.0	99	99
343	135	4.01	99.0	60.0	161.0	90.0	1111	99
344	137	3.90	30.0	21.0	89.0	12.0	99	99
345	99	99.00	53.0	99.0	99.0	99.0	99	1
346	141	4.60	83.6	34.3	71.5	5.6	99	99

[328 rows x 172 columns]

```
[40]: #Convert values called '99' to missing values
test_data.replace(99, pd.NA, inplace = True)
test_data.replace('99', pd.NA, inplace = True)
test_data.isnull().sum()
```

```
[40]: Hospital      0
Sample          0
ICU             7
OPD             7
Sex            17
...
T.Bilirubin     245
S.creatinine    205
B.urea          214
ALP            265
Puscells       266
Length: 172, dtype: int64
```

```
[41]: #Adjust data types in test data
test_data[numerical_vars] = test_data[numerical_vars].apply(pd.to_numeric,
↳errors = 'coerce')
```

```
[42]: #Impute numerical variables with median
for col in numerical_vars:
    if test_data[col].isnull().sum() > 0:
        median_val = test_data[col].median()
        test_data[col].fillna(median_val, inplace = True)
```

```
[43]: #Perform robust scaling on numerical data
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()
test_data[numerical_vars] = scaler.fit_transform(test_data[numerical_vars])
test_data[numerical_vars] = pd.DataFrame(test_data[numerical_vars], columns =
    ↪numerical_vars)
test_data[numerical_vars]
```

```
[43]:
```

	Age	PRad	SBPadd	WBCcount	Ncount	N	Lcount	\
0	0.166667	0.0	0.0	2.889064	17.862857	2.879831	11.000000	
1	0.055556	0.0	0.0	-1.973249	-8.556190	-6.815778	-15.333333	
2	0.277778	0.0	0.0	0.000000	0.000000	0.000000	0.000000	
3	-0.500000	0.0	0.0	1.117231	8.986667	3.381831	-0.333333	
4	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	
..	
342	-1.055556	6.0	0.0	2.077105	-7.769524	-18.755004	25.666667	
343	1.722222	2.0	-30.0	0.084972	3.196190	2.878263	-5.666667	
344	-0.500000	-18.0	0.0	0.361920	3.900952	1.665318	-6.000000	
345	-0.777778	-8.0	-10.0	-0.242329	0.000000	0.000000	0.000000	
346	0.444444	32.0	-40.0	-0.484658	0.262857	2.985140	-10.666667	

	L	Plateletcount	PCV	...	ESR	Redcells	Na	K	AST	\
0	-2.442451	-5.534884	-0.8	...	0.0	0.0	0.0	0.00	0.0	
1	8.526969	26.930233	-0.8	...	0.0	0.0	0.0	0.00	127.0	
2	0.000000	0.000000	0.0	...	0.0	0.0	0.0	0.00	0.0	
3	-2.095642	5.441860	2.2	...	0.0	0.0	0.0	0.00	0.0	
4	0.000000	0.000000	0.0	...	0.0	0.0	0.0	0.00	0.0	
..	
342	-0.726531	-10.186047	3.2	...	0.0	0.0	8.0	-0.60	-26.3	
343	-1.487197	-9.720930	-4.8	...	0.0	0.0	0.0	0.01	0.0	
344	-1.922303	-0.790698	2.7	...	0.0	0.0	2.0	-0.10	-20.0	
345	0.000000	0.000000	0.0	...	0.0	10.0	0.0	0.00	3.0	
346	-1.256950	-6.558140	-14.0	...	0.0	0.0	6.0	0.60	33.6	

	T.Bilirubin	S.creatinine	B.urea	ALP	Puscells
0	0.0	0.0	0.0	0.0	0.0
1	0.0	-13.6	-4.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0
..
342	-7.0	-37.0	0.0	0.0	0.0
343	44.8	59.0	78.0	1030.0	0.0
344	5.8	-13.0	0.0	0.0	0.0
345	0.0	0.0	0.0	0.0	-1.0
346	19.1	-30.5	-6.4	0.0	0.0

[328 rows x 22 columns]

```
[44]: #Impute categorical variables with mode
      for col in categorical_vars_filtered:
          test_data[col].fillna(test_data[col].mode()[0], inplace = True)
```

```
[45]: #Verification
      test_data.isnull().sum()
```

```
[45]: Hospital      0
      Sample      0
      ICU         0
      OPD         0
      Sex         0
      ..
      T.Bilirubin  0
      S.creatinine 0
      B.urea      0
      ALP        0
      Puscells    0
      Length: 172, dtype: int64
```

```
[46]: #Create a new csv after test data preprocessing if needed
      test_data.to_csv('Preprocessed_test_RF.csv', index = False)
      test_data.shape
```

```
[46]: (328, 172)
```

1.28 Handle column discrepancy between train and test data

```
[47]: #Check if both train and test data have the same columns
      train_Final_dropped = train_data.columns.drop('Final')
      missing_cols_test = set(train_Final_dropped) - set(test_data.columns)
      print("Columns in train but not in test:", missing_cols_test)
```

Columns in train but not in test: set()

```
[48]: #Ensure the column order is the same as in the train data
      test_data = test_data[train_Final_dropped]
```

1.29 Predict test data

```
[49]: y_pred = best_model.predict(test_data)
      print(y_pred.astype(int))
```

```
[1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1
 2 2 2 2 2 1 1 1 2 2 2 2 1 2 2 1 1 1 2 2 2 2 1 2 2 1 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 1 2 2 2 2 1 2 2 1 2 2 1
 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 1 2 2 2 1 1 1 1 2 1 1 1 1 2 2 2 2 2 2 2
 2 2 2 2 1 1 1 1 1 1 1 2 1 2 1 1 1 1 2 2 1 2 2 2 2 2 2 2 1 1 1 1 1 2 2 1 1 1 2
 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 1 2 2 1 2 2 1 2 2 2 1 2 2 2 1 1 1
 1 1 1 1 2 2 2 2 2 2 2 2 1 2 1 1 1 2 2 2 1 2 2 1 1 2 1 1 1 2 2 2 2 2 1 2 2
 1 2 1 1 1 2 2 1 1 2 1 2 2 2 2 2 2 2 2 1 2 1 2 1 2 1 1 2 2 2 2 2 1 1 2 2 2
 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 2 2 2 2 2 1 2 2 1 2]
```

```
[50]: #Preparing submission file - Logistic Regression
submission = pd.DataFrame({'ID': range(1, len(y_pred) + 1), 'Final': y_pred})
submission.to_csv('Predictions_RF.csv', index = False)
```