In [1]:
```
#  import required packages
from scipy.stats import chi2_contingency
from matplotlib.lines import Line2D
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import json
import sklearn
from sklearn.preprocessing import LabelEncoder
import networkx as nx
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
from matplotlib.pylab import rcParams
from itertools import combinations
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
class PredictaVie_Preprocess:
    def __init__(self, dataframe):
        self.dataframe = dataframe

    def event_pie_chart(self, column_name):
        counts = self.dataframe[column_name].value_counts()
        total_count = counts.sum()
        ratios = counts / total_count
        significant_events = ratios[ratios > 0.01].index.tolist()
        other_ratio = ratios[ratios <= 0.01].sum()
        plt.figure(figsize=(8, 6))
        plt.pie(ratios[ratios > 0.01].values.tolist() + [other_ratio], labels=si
        plt.title('Distribution of ' + column_name + ' in the target dataset:')
        plt.show()

    def filter_data_by_significant_events(self, column_name):
        significant_events = self.dataframe[column_name].value_counts(normalize=
        significant_events_data = self.dataframe[self.dataframe[column_name].isi
        self.filtered = significant_events_data
        return significant_events_data


    def ind_event_sequence(self, person_id):
        df = self.dataframe
        df['event_date'] = pd.to_datetime(df['event_date'])
        df0 = df[df['person_id'] == person_id]
        df1 = df0.sort_values(by='event_date')
        df2 = df1[df1['event'] != df1['event'].shift(-1)]
        df2.reset_index(drop=True, inplace=True)
        return df2

    def plot_journey(self, person_id):
        df = self.dataframe
        df['event_date'] = pd.to_datetime(df['event_date'])
        person_data = df[df['person_id'] == person_id]
```

```
        person_data_sorted = person_data.sort_values(by='event_date')
        person_data_unique = person_data_sorted[person_data_sorted['event'] != p
        person_data_unique.reset_index(drop=True, inplace=True)
        G = nx.DiGraph()
        for _, row in person_data_unique.iterrows():
            G.add_node(row['event'])
        for i in range(len(person_data_unique) - 1):
            current_event = person_data_unique.iloc[i]['event']
            next_event = person_data_unique.iloc[i + 1]['event']
            G.add_edge(current_event, next_event)
        pos = nx.circular_layout(G)
        nx.draw(G, pos, with_labels=True, node_size=2000, node_color='skyblue',
                arrows=True, arrowsize=20)
        plt.title('Event Sequence for Person ID: ' + str(person_id))
        plt.show()

    def create_event_sequence(self):
        if not hasattr(self, 'filtered'):
            raise AttributeError("The 'filtered' data has not been generated. Pl

        event_sequence_df = pd.DataFrame(columns=['person_id', 'gender'])
        last_event_df = pd.DataFrame(columns=['person_id'])
        for person_id, group in self.filtered.groupby('person_id'):
            gender = group['gender'].iloc[0]
            events = group.sort_values(by='event_date')['event'].tolist()
            previous_event = None
            event_list = []
            for event in events:
                if event != previous_event:
                    event_list.append(event)
                previous_event = event
            event_sequence = {'person_id': person_id, 'gender': gender}
            sequence_count = 0
            for i in event_list[:-1]:
                sequence_count += 1
                event_sequence[f'condition_{sequence_count}'] = i
            event_sequence_df = pd.concat([event_sequence_df, pd.DataFrame(event
                                           ignore_index=True)
            last_event_df = pd.concat([last_event_df, pd.DataFrame({'person_id':
                                                                    'last_condit
                                       ignore_index=True)
        all_together = pd.merge(event_sequence_df, last_event_df, on='person_id'
        self.event_sequence = all_together
        return all_together

    def filter_long_journey(self, top_numbers=2000):
        if not hasattr(self, 'event_sequence'):
            raise AttributeError("The 'event_sequence' attribute has not been ge

        data = self.event_sequence
        data['null_count'] = data.isnull().sum(axis=1)
        sort = data.sort_values(by='null_count', ascending=True)
        sort.reset_index(drop=True, inplace=True)
        result = sort.iloc[:, :-1].head(top_numbers)
        self.long_journey = result
        return result

    def generate_pairs(self):
        if not hasattr(self, 'long_journey'):
            raise AttributeError("The 'long_journey' attribute has not been gene
```

```
        df = self.long_journey
        pairs = []
        for index, row in df.iterrows():
            person_id = row['person_id']
            gender = row['gender']
            last = row['last_condition']
            conditions = [col for col in row['condition_1':'last_condition'] if

            for pair in combinations(conditions, 2):
                if pair[0] != pair[1]:  # avoid having same conditions paired
                    pair_str = ' -> '.join(pair)
                    pair_str = pair_str.replace('[', '').replace(']', '')
                    pairs.append({'person_id': person_id, 'gender': gender, 'pai
                                  'last_condition': last})

        pairs_df = pd.DataFrame(pairs)
        self.pairs = pairs_df
        return pairs_df

    def pairs_to_bin(self):
        if not hasattr(self, 'pairs'):
            raise AttributeError("The 'pairs' attribute has not been generated.

        df = self.pairs
        pair_type = df['pair'].unique()
        pair_bin_df = pd.DataFrame(columns=['person_id', 'MALE', 'FEMALE'] + lis

        for person_id, group in df.groupby('person_id'):
            gender = group['gender'].iloc[0]
            last = group['last_condition'].iloc[0]
            pair_bin = {'person_id': person_id, 'last_condition': last}

            for i in ['MALE', 'FEMALE']:
                pair_bin[i] = 0
                if i == gender:
                    pair_bin[i] = 1

            for i in pair_type:
                pair_bin[i] = 0
                if i in group['pair'].values:
                    pair_bin[i] = 1

            pair_bin_df = pd.concat([pair_bin_df, pd.DataFrame(pair_bin, index=[

        return pair_bin_df

    def pairs_to_count(self):
        if not hasattr(self, 'pairs'):
            raise AttributeError("The 'pairs' attribute has not been generated.

        df = self.pairs
        pair_types = df['pair'].unique()
        pair_count_df = pd.DataFrame(columns=['person_id', 'MALE', 'FEMALE'] + l

        for person_id, group in df.groupby('person_id'):
            gender = group['gender'].iloc[0]
            last = group['last_condition'].iloc[0]

            pair_counts = group['pair'].value_counts().to_dict()
```

```
            pair_count = {'person_id': person_id, 'last_condition': last}

            for i in ['MALE', 'FEMALE']:
                pair_count[i] = 0
                if i == gender:
                    pair_count[i] = 1

            for pair_type in pair_types:
                pair_count[pair_type] = pair_counts.get(pair_type, 0)

            pair_count_df = pd.concat([pair_count_df, pd.DataFrame([pair_count])

        return pair_count_df

class PredictaVie_SplitData:
    def __init__(self, dataframe):
        self.dataframe = dataframe

    def c2c_split_data(self, test_size=0.2, random_state=42):
        df = self.dataframe
        exclude_column = ['person_id', 'last_condition']
        filtered_columns = [col for col in df.columns if col not in exclude_colu
        y = df['last_condition']
        x = df.loc[:, filtered_columns].astype(int)
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test
        return x_train, x_test, y_train, y_test

class PredictaVie_Model:
    def __init__(self, x_train, y_train, x_test, y_test):
        self.x_train = x_train
        self.y_train = y_train
        self.x_test = x_test
        self.y_test = y_test

    def c2c_train_xgb(self, param_grid=None, cv=5):
        label_encoder = LabelEncoder()
        y_train_encoded = label_encoder.fit_transform(self.y_train)

        if param_grid is None:
            param_grid = {
                'n_estimators': [100, 150, 200],
                'max_depth': [3, 4, 5],
                'learning_rate': [0.1, 0.01, 0.001]
            }

        model = xgb.XGBClassifier()
        grid_search = GridSearchCV(model, param_grid, cv=cv, scoring='accuracy')
        grid_search.fit(self.x_train, y_train_encoded)

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

        print("Best parameters:", best_params)
        print("Best cross-validation accuracy:", best_score)

        self.best_xgbmodel = best_model

        return best_model
```

```
    def c2c_evaluate_xgb(self):
        if not hasattr(self, 'best_xgbmodel'):
            raise AttributeError("The 'best_xgbmodel' has not been generated. Pl

        best = self.best_xgbmodel

        label_encoder = LabelEncoder()
        y_test_encoded = label_encoder.fit_transform(self.y_test)

        y_pred = best.predict(self.x_test)

        accuracy = accuracy_score(y_test_encoded, y_pred)
        print("Test Accuracy:", accuracy)

        original_labels = label_encoder.inverse_transform(y_test_encoded)
        predicted_labels = label_encoder.inverse_transform(y_pred)
        cm = confusion_matrix(original_labels, predicted_labels)

        cm_df = pd.DataFrame(cm, index=label_encoder.classes_, columns=label_enc

        plt.figure(figsize=(10, 8))
        sns.heatmap(cm_df, annot=True, cmap="YlGnBu", fmt="d")
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.title('Confusion Matrix')
        plt.show()

    def c2c_xgb_feature_importance(self):
        if not hasattr(self, 'best_xgbmodel'):
            raise AttributeError("The 'best_xgbmodel' has not been generated. Pl

        xgb_model = self.best_xgbmodel
        feature_importance = xgb_model.feature_importances_
        feature_names = xgb_model.get_booster().feature_names
        feature_importance_dict = dict(zip(feature_names, feature_importance))
        sorted_feature_importance = sorted(feature_importance_dict.items(), key=

        features = [x[0] for x in sorted_feature_importance[:20]]
        importance = [x[1] for x in sorted_feature_importance[:20]]

        plt.figure(figsize=(10, 8))
        plt.barh(features, importance, color='skyblue')
        plt.xlabel('Feature Importance')
        plt.ylabel('Features')
        plt.title('XGBoost Feature Importance')
        plt.gca().invert_yaxis()
        plt.show()
        top_20_feature_indices = [item[0] for item in sorted_feature_importance[
        return top_20_feature_indices

    def c2c_train_logreg(self, param_grid=None):
        logistic_reg = LogisticRegression()

        if param_grid is None:
            param_grid = {
                'penalty': ['l1', 'l2', 'elasticnet', 'none'],
                'C': [0.001, 0.01, 0.1, 1, 10, 100],
                'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
            }
```

```
        grid_search = GridSearchCV(estimator=logistic_reg, param_grid=param_grid
        grid_search.fit(self.x_train, self.y_train)
        print("Best Parameters:", grid_search.best_params_)

        best_model = grid_search.best_estimator_

        feature_names = list(self.x_train.columns)
        feature_weights = best_model.coef_[0]
        feature_weight_dict = dict(zip(feature_names, feature_weights))
        print("\nFeature Weights in the Best Model:")
        for feature, weight in feature_weight_dict.items():
            print(f"{feature}: {weight}")

        abs_feature_weights = np.abs(feature_weights)
        top_20_indices = np.argsort(abs_feature_weights)[::-1][:20]
        top_20_features = [feature_names[i] for i in top_20_indices]

        self.best_logregmodel = best_model

        return best_model, top_20_features

    def c2c_evaluate_logreg(self):
        if not hasattr(self, 'best_logregmodel'):
            raise AttributeError("The 'best_logregmodel' has not been generated.

        best = self.best_logregmodel

        y_pred = best.predict(self.x_test)

        accuracy = accuracy_score(self.y_test, y_pred)
        print("Test Accuracy:", accuracy)

        cm = confusion_matrix(self.y_test, y_pred)
        cm_df = pd.DataFrame(cm, index=best.classes_, columns=best.classes_)

        plt.figure(figsize=(10, 8))
        sns.heatmap(cm_df, annot=True, cmap="Blues", fmt='g')
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.title('Confusion Matrix Heatmap')
        plt.show()
```

In [3]:

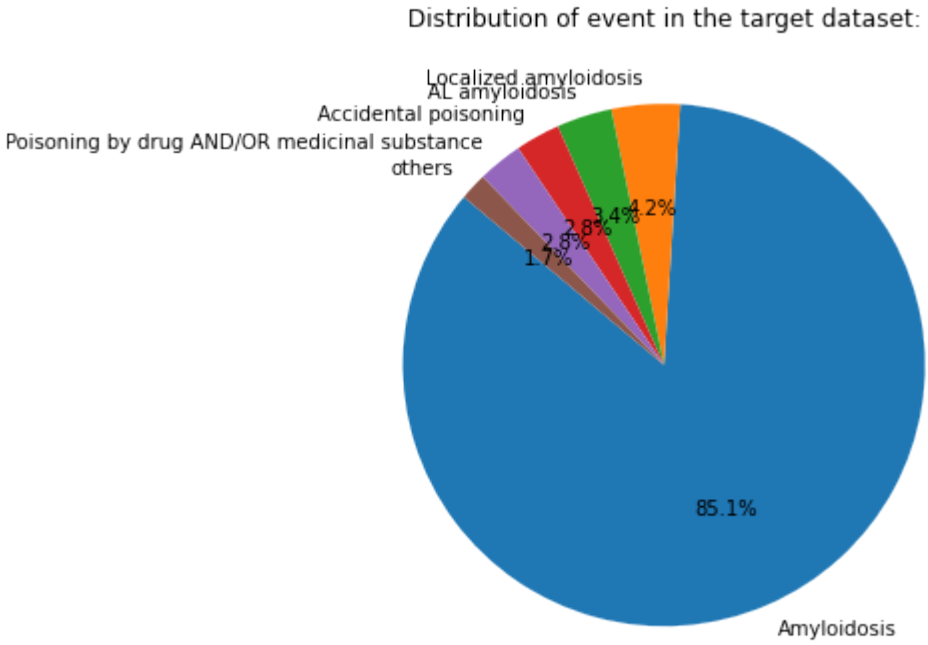
```
# Load the Anxiety event data
from dbconns import ImpalaConnector
ic = ImpalaConnector('CHUNGAX6','Francais1418!', connection = "arch-prod-impala-

data_amyloidosis = ic.read("SELECT * FROM app_rwd_capstone_group3.data_amyloidos
data_amyloidosis.head()
```

Out[3]:

	person_id	age_when_event	gender	event	event_date
0	39602	64	FEMALE	Amyloidosis	2007-08-17
1	39602	64	FEMALE	Amyloidosis	2007-05-11
2	39602	64	FEMALE	Amyloidosis	2007-08-17
3	39602	64	FEMALE	Amyloidosis	2007-08-10
4	39602	64	FEMALE	Amyloidosis	2007-05-23

In [4]:
```
# load the preprocess package
preprocess = PredictaVie_Preprocess(data_amyloidosis)
```

In [5]:
```
# show the distribution of each events in the dataset
preprocess.event_pie_chart('event')
```



In [6]:
```
# filter those insignificant events for smoother model building
preprocess.filter_data_by_significant_events('event')
```

Out[6]:

	person_id	age_when_event	gender	event	event_date
0	39602	64	FEMALE	Amyloidosis	2007-08-17
1	39602	64	FEMALE	Amyloidosis	2007-05-11
2	39602	64	FEMALE	Amyloidosis	2007-08-17
3	39602	64	FEMALE	Amyloidosis	2007-08-10
4	39602	64	FEMALE	Amyloidosis	2007-05-23
...	...	...	...	...	...
9995	222619701	63	MALE	AL amyloidosis	2020-10-19
9996	222619701	62	MALE	Amyloidosis	2019-04-02
9997	222619701	62	MALE	Amyloidosis	2019-03-04
9998	222619701	62	MALE	Amyloidosis	2019-03-10
9999	222619701	61	MALE	Amyloidosis	2018-10-18

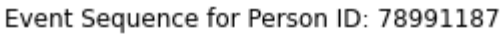
9831 rows × 5 columns

In [7]:
```
# take "patient 78991187" as an example
preprocess.ind_event_sequence(78991187)
```

Out[7]:

person_id	age_when_event	gender	event	event_date

In [8]:
```
# pateint 78991187's condition journey
preprocess.plot_journey(78991187)
```



In [9]:
```
# create the condition sequence
preprocess.create_event_sequence()
```

Out[9]:

	person_id	gender	condition_1	condition_2	condition_3	condition_4	condition_
0	39602	FEMALE	NaN	NaN	NaN	NaN	Na
1	3742902	FEMALE	NaN	NaN	NaN	NaN	Na
2	4353902	FEMALE	NaN	NaN	NaN	NaN	Na
3	4939102	FEMALE	NaN	NaN	NaN	NaN	Na
4	5754701	MALE	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	
989	222136701	MALE	NaN	NaN	NaN	NaN	Na
990	222192702	FEMALE	NaN	NaN	NaN	NaN	Na
991	222218801	MALE	NaN	NaN	NaN	NaN	Na
992	222459501	MALE	NaN	NaN	NaN	NaN	Na
993	222619701	MALE	Amyloidosis	AL amyloidosis	Amyloidosis	AL amyloidosis	Amyloidos

994 rows × 59 columns

In [10]:
```
# take the top 2000 patients who have long condition journey (lots of event hist
preprocess.filter_long_journey()
```

Out[10]:

	person_id	gender	condition_1	condition_2	condition_3	condition_4	condition_
0	222619701	MALE	Amyloidosis	AL amyloidosis	Amyloidosis	AL amyloidosis	Amyloidos
1	172051501	MALE	Amyloidosis	Localized amyloidosis	Amyloidosis	Localized amyloidosis	Amyloidos
2	208433401	FEMALE	Localized amyloidosis	Amyloidosis	Localized amyloidosis	Amyloidosis	Localize amyloidos
3	141537602	MALE	Localized amyloidosis	Amyloidosis	Localized amyloidosis	Amyloidosis	Localize amyloidos
4	133243601	MALE	Amyloidosis	AL amyloidosis	Amyloidosis	AL amyloidosis	Amyloidos
...	...	...	...	...	...	...	
989	15795103	FEMALE	NaN	NaN	NaN	NaN	Na
990	15800402	FEMALE	NaN	NaN	NaN	NaN	Na
991	16020301	MALE	NaN	NaN	NaN	NaN	Na
992	16224903	FEMALE	NaN	NaN	NaN	NaN	Na
993	39602	FEMALE	NaN	NaN	NaN	NaN	Na

994 rows × 59 columns

In [11]:
```
# create condition pairs on the filtered data (including duplicated pairs)
preprocess.generate_pairs()
```

Out[11]:

	person_id	gender	pair	last_condition
0	222619701	MALE	Amyloidosis -> AL amyloidosis	AL amyloidosis
1	222619701	MALE	Amyloidosis -> AL amyloidosis	AL amyloidosis
2	222619701	MALE	Amyloidosis -> AL amyloidosis	AL amyloidosis
3	222619701	MALE	Amyloidosis -> AL amyloidosis	AL amyloidosis
4	222619701	MALE	Amyloidosis -> Localized amyloidosis	AL amyloidosis
...	...	...	...	...
5937	218198001	FEMALE	Accidental poisoning -> Poisoning by drug AND/...	Poisoning by drug AND/OR medicinal substance
5938	218373803	MALE	Accidental poisoning -> Poisoning by drug AND/...	Poisoning by drug AND/OR medicinal substance
5939	218451301	FEMALE	Accidental poisoning -> Poisoning by drug AND/...	Poisoning by drug AND/OR medicinal substance
5940	219006403	MALE	Accidental poisoning -> Poisoning by drug AND/...	Poisoning by drug AND/OR medicinal substance
5941	16144201	FEMALE	Accidental poisoning -> Poisoning by drug AND/...	Poisoning by drug AND/OR medicinal substance

5942 rows × 4 columns

In [12]:
```
# make the condition pairs into count
paircount = preprocess.pairs_to_count()
paircount
```

Out[12]:

	person_id	MALE	FEMALE	Amyloidosis -> AL amyloidosis	Amyloidosis -> Localized amyloidosis	AL amyloidosis -> Amyloidosis	AL amyloidosis -> Localized amyloidosis
0	15195302	0	1	0	15	0	0
1	15494901	1	0	3	1	3	1
2	15892302	1	0	36	0	36	0
3	16144201	0	1	0	0	0	0
4	21235701	1	0	0	0	0	0
...	...	...	...	...	...	...	...
269	218198001	0	1	0	0	0	0
270	218373803	1	0	0	0	0	0
271	218451301	0	1	0	0	0	0
272	219006403	1	0	0	0	0	0
273	222619701	1	0	338	108	262	93

274 rows × 12 columns

In [13]:
```
# make the condition pairs into binary
pairbin = preprocess.pairs_to_bin()
pairbin
```

Out[13]:

	person_id	MALE	FEMALE	Amyloidosis -> AL amyloidosis	Amyloidosis -> Localized amyloidosis	AL amyloidosis -> Amyloidosis	AL amyloidosis -> Localized amyloidosis
0	15195302	0	1	0	1	0	0
1	15494901	1	0	1	1	1	1
2	15892302	1	0	1	0	1	0
3	16144201	0	1	0	0	0	0
4	21235701	1	0	0	0	0	0
...	...	...	...	...	...	...	...
269	218198001	0	1	0	0	0	0
270	218373803	1	0	0	0	0	0
271	218451301	0	1	0	0	0	0
272	219006403	1	0	0	0	0	0
273	222619701	1	0	1	1	1	1

274 rows × 12 columns

In [14]:
```
# load the train_test_split package
paircountsplit = PredictaVie_SplitData(paircount)
pairbinsplit = PredictaVie_SplitData(pairbin)
```

In [15]:
```
# Split data into training and testing, respectively
x_train, x_test, y_train, y_test = paircountsplit.c2c_split_data() # for count a
X_train, X_test, Y_train, Y_test = pairbinsplit.c2c_split_data() # for binary da
```

In [16]:
```
# load the model package
countmodel = PredictaVie_Model(x_train, y_train, x_test, y_test)
binmodel = PredictaVie_Model(X_train, Y_train, X_test, Y_test)
```

In [17]:
```
# Use Gridsearch to build and find the best XGBoost model
binmodel.c2c_train_xgb()
```

```
Best parameters: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 100}
Best cross-validation accuracy: 0.9132135306553911
```

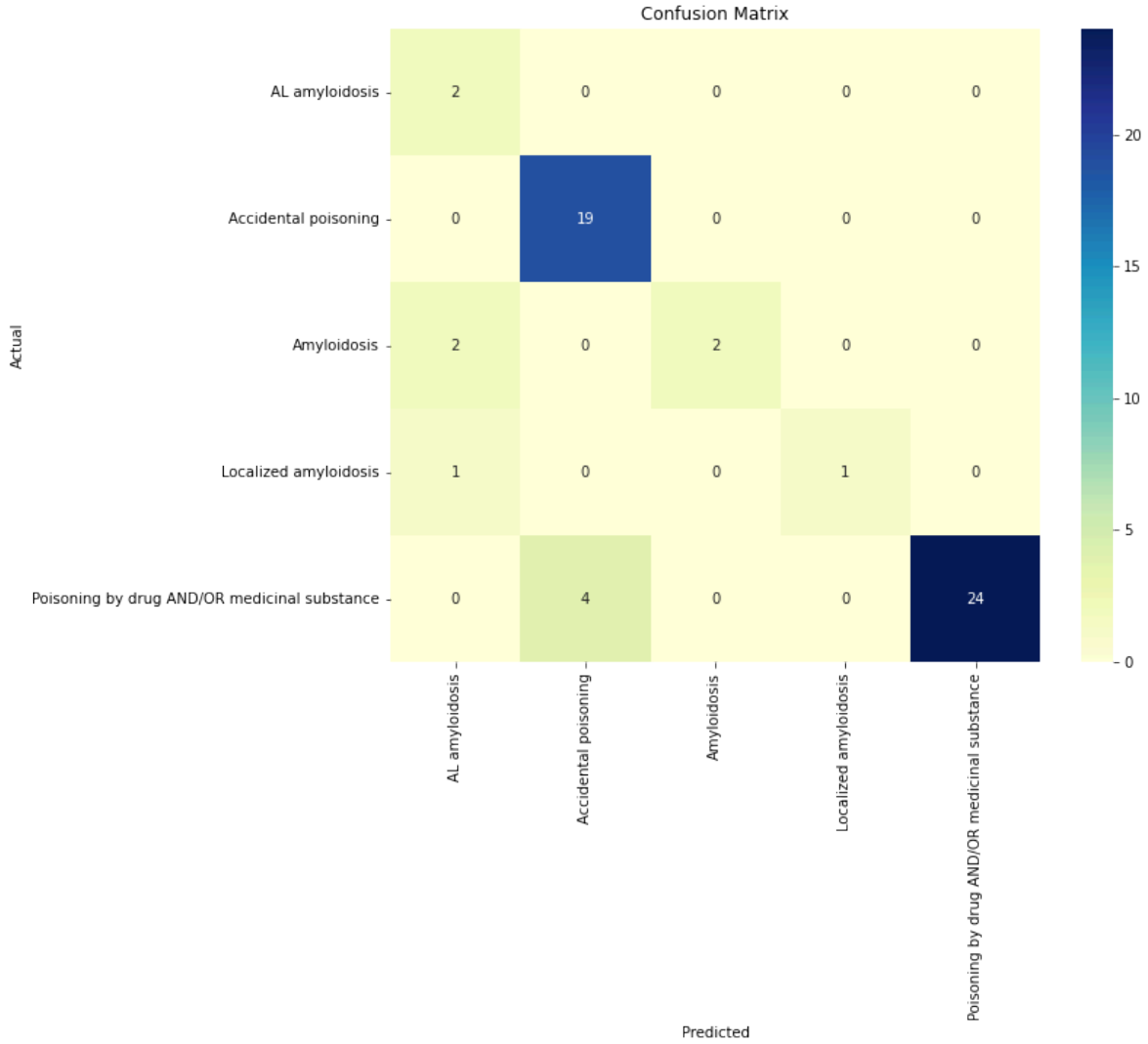
Out[17]:

XGBClassifier

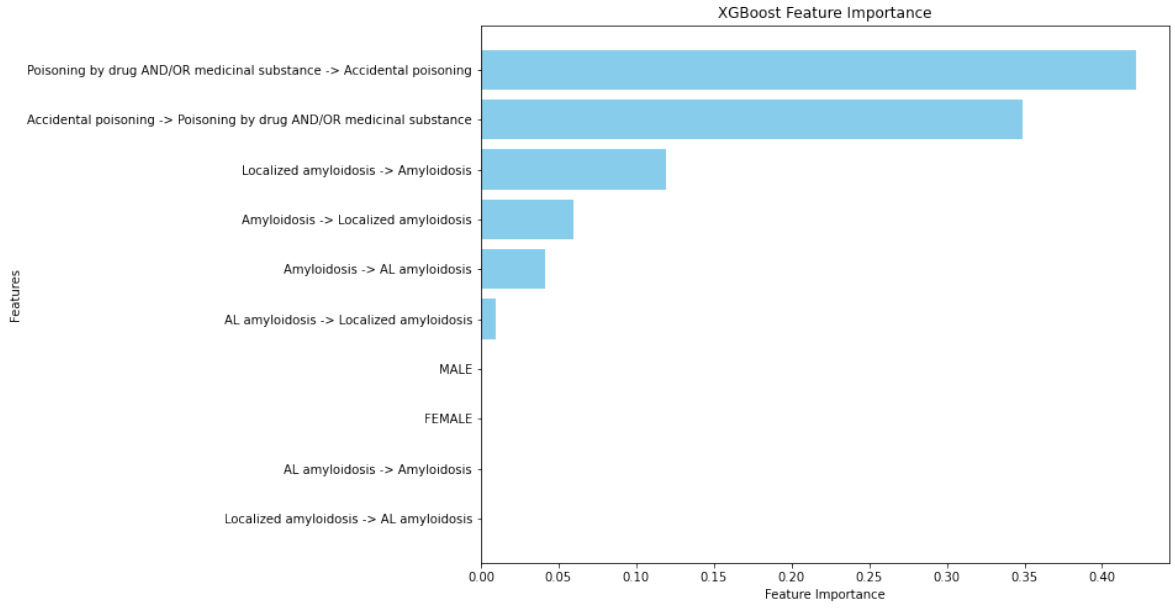
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rou
nds=None,
              enable_categorical=False, eval_metric=None, feature_ty
pes=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.01, max_
bin=None,
```

In [18]:
```
# test the best XGBoost model to see its performance
binmodel.c2c_evaluate_xgb()
```

```
Test Accuracy: 0.8727272727272727
```



In [19]:
```
# Check which condtion pairs are important for predictions
xgb_top_features = binmodel.c2c_xgb_feature_importance()
xgb_top_features # store the top 20 important features into a list
```



Out[19]:
```
['Poisoning by drug AND/OR medicinal substance -> Accidental poisoning',
 'Accidental poisoning -> Poisoning by drug AND/OR medicinal substance',
 'Localized amyloidosis -> Amyloidosis',
 'Amyloidosis -> Localized amyloidosis',
 'Amyloidosis -> AL amyloidosis',
 'AL amyloidosis -> Localized amyloidosis',
 'MALE',
 'FEMALE',
 'AL amyloidosis -> Amyloidosis',
 'Localized amyloidosis -> AL amyloidosis']
```

In [20]:
```
# ignore "convergence not found" warnings while the gridsearch is running
from sklearn.exceptions import ConvergenceWarning
import warnings
warnings.filterwarnings("ignore", category=ConvergenceWarning)


# Use Gridsearch to build and find the best Logistic Regression model
_, logreg_top_features = countmodel.c2c_train_logreg()
logreg_top_features # store the top 20 important features into a list
```

```
Best Parameters: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}

Feature Weights in the Best Model:
MALE: -0.4668388374694927
FEMALE: -1.3058894719235985
Amyloidosis -> AL amyloidosis: 1.2302676888849573
Amyloidosis -> Localized amyloidosis: 0.7145268937626458
AL amyloidosis -> Amyloidosis: 0.9428136167509052
AL amyloidosis -> Localized amyloidosis: -0.147472274641745
Localized amyloidosis -> Amyloidosis: 2.1886953451256757
Localized amyloidosis -> AL amyloidosis: 0.24573061761817683
Poisoning by drug AND/OR medicinal substance -> Accidental poisoning: -1.08828995
57724106
Accidental poisoning -> Poisoning by drug AND/OR medicinal substance: -1.14334663
92874532
```

```
/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_validatio
n.py:547: FitFailedWarning:
390 fits failed out of a total of 600.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_sc
ore='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_va
lidation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/base.py", line 147
4, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 67, in _check_solver
    raise ValueError(
ValueError: Solver newton-cg supports only 'l2' or None penalties, got l1 penalt
y.

--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_va
lidation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/base.py", line 147
4, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 67, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or None penalties, got l1 penalty.

--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_va
lidation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/base.py", line 147
4, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 67, in _check_solver
    raise ValueError(
ValueError: Solver sag supports only 'l2' or None penalties, got l1 penalty.
```

```
--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_va
lidation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/base.py", line 147
4, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 67, in _check_solver
    raise ValueError(
ValueError: Solver newton-cg supports only 'l2' or None penalties, got elasticnet
penalty.

--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_va
lidation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/base.py", line 147
4, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 67, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or None penalties, got elasticnet pen
alty.

--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_va
lidation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/base.py", line 147
4, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 75, in _check_solver
    raise ValueError(
ValueError: Only 'saga' solver supports elasticnet penalty, got solver=liblinear.

--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_va
lidation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/base.py", line 147
```

```
4, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 1172, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 67, in _check_solver
    raise ValueError(
ValueError: Solver sag supports only 'l2' or None penalties, got elasticnet penal
ty.

--------------------------------------------------------------------------------
30 fits failed with the following error:
Traceback (most recent call last):
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_va
lidation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/base.py", line 147
4, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/linear_model/_logis
tic.py", line 1182, in fit
    raise ValueError("l1_ratio must be specified when penalty is elasticnet.")
ValueError: l1_ratio must be specified when penalty is elasticnet.

--------------------------------------------------------------------------------
150 fits failed with the following error:
Traceback (most recent call last):
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_va
lidation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/base.py", line 146
7, in wrapper
    estimator._validate_params()
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/base.py", line 666,
in _validate_params
    validate_parameter_constraints(
  File "/home/cdsw/.local/lib/python3.9/site-packages/sklearn/utils/_param_valida
tion.py", line 95, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'penalty' parameter of
LogisticRegression must be a str among {'l1', 'elasticnet', 'l2'} or None. Got 'n
one' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/home/cdsw/.local/lib/python3.9/site-packages/sklearn/model_selection/_search.py:
1051: UserWarning: One or more of the test scores are non-finite: [       nan
nan 0.03646934        nan 0.43837209 0.52061311
 0.52061311 0.87209302 0.87209302 0.87209302        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan 0.03646934        nan
 0.48393235 0.87209302 0.87663848 0.88128964 0.87674419 0.87209302
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
 0.84936575        nan 0.87209302 0.88128964 0.88128964 0.8859408
 0.87674419 0.87209302        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan 0.92241015        nan 0.87209302 0.91786469
 0.92241015 0.92241015 0.87674419 0.87209302        nan        nan
        nan        nan        nan        nan        nan        nan
```

```
       nan        nan        nan        nan 0.92241015        nan
0.87209302 0.9269556  0.92241015 0.92241015 0.87674419 0.87209302
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
0.91331924        nan 0.87209302 0.91321353 0.93604651 0.92241015
0.87674419 0.87209302        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan]
 warnings.warn(
```

Out[20]:
```
['Localized amyloidosis -> Amyloidosis',
 'FEMALE',
 'Amyloidosis -> AL amyloidosis',
 'Accidental poisoning -> Poisoning by drug AND/OR medicinal substance',
 'Poisoning by drug AND/OR medicinal substance -> Accidental poisoning',
 'AL amyloidosis -> Amyloidosis',
 'Amyloidosis -> Localized amyloidosis',
 'MALE',
 'Localized amyloidosis -> AL amyloidosis',
 'AL amyloidosis -> Localized amyloidosis']
```

In [21]:
```
# test the best Logistic Regression model to see its performance
countmodel.c2c_evaluate_logreg()
```

```
Test Accuracy: 0.9636363636363636
```