

Relazione progetto: Heart Disease and Stroke prevention

Emanuele Conforti - 252122

Jacopo Garofalo - 252093

Gianmarco La Marca - 252256

May 2024

Indice

1	Introduzione	2
1.1	Business understanding	2
2	Data understanding	3
2.1	Analisi preliminare del dataset	3
2.2	Valori 'null'	7
2.3	Eliminazione degli attributi strettamente correlati	8
2.4	Conversione degli attributi di tipo object in categorici	9
2.5	Iistogrammi e grafici	10
3	Data preparation	15
3.1	Eliminazione degli ultimi attributi	15
3.2	Riempimento dei valori 'null'	16
3.3	Binarizzazione degli attributi	18
4	Modeling	20
4.1	Suddivisione del Data Set	20
4.2	Modelli	21
5	Best model evaluation	26
5.0.1	Valutazione del best model sul test set	26
5.0.2	ROC curve	27
6	Conclusioni	30

Capitolo 1

Introduzione

1.1 Business understanding

Il dataset **Heart Disease and Stroke Prevention** contiene dati relativi alla prevenzione delle malattie cardiache e degli ictus; include varie metriche e indicatori per analizzare e prevedere l'insorgenza di queste condizioni. Consiste in dati clinici raccolti dal CMS (*Center for Medicare and Medicaid Services*) di pazienti ospedalizzati e ambulatoriali. Il sistema è progettato per integrare molteplici indicatori da numerose fonti di dati per fornire un quadro completo sulle malattie cardiovascolari e dei fattori di rischio associati negli Stati Uniti. I dati sono organizzati per località (nazionale e statale) e indicatore e possono essere stratificati per sesso ed etnia.

Il dataset è progettato per aiutare a identificare fattori di rischio riguardanti le malattie cardiovascolari e sviluppare modelli predittivi per queste condizioni. Infatti, l'obiettivo del seguente lavoro è stato quello di definire una procedura automatica per la categorizzazione delle malattie cardiovascolari.

Capitolo 2

Data understanding

2.1 Analisi preliminare del dataset

Il dataset preso in considerazione contiene dati relativi ai pazienti che hanno partecipato ai programmi sanitari pubblici statunitensi Medicare e Medicaid per diverse categorie e anni. In particolare, il dataset è composto da 42640 righe, ognuna delle quali rappresenta una diversa visita, e 29 colonne, di cui 28 sono attributi e 1 rappresenta la nostra **target value**, il *Topic*.

Il dataset è stato diffuso assieme ad una descrizione per ognuna delle colonne presenti in esso.

Tabella 2.1: Proprietà degli attributi

Attributo	Descrizione
Year	Anno in cui si è effettuata la visita
LocationAbbr	Abbreviazione della località secondo il sistema statunitense
LocationDesc	Nome completo della località
DataSource	Abbreviazione della fonte del dato corrente
PriorityArea1	Area di priorità (Million Hearts® o None)
PriorityArea2	Area di priorità (ABCS o None)
PriorityArea3	Area di priorità (Healthy People 2020 o None)
PriorityArea4	Area di priorità (AHA 2020 Goals: Cardiovascular Health Metrics o None)
Category	Descrizione della categoria di malattie per la visita corrente
Topic	Descrizione dell'oggetto della visita

Continua alla pagina successiva

Tabella 2.1 – continua dalla pagina precedente

Attributo	Descrizione
Indicator	Descrizione dell'indicatore sulla visita
Data_Value_type	Tipo di valore del dato (media, tasso, percentuale)
Data_Value_Unit	Unità del valore del dato (% , tasso su 100,000,etc.)
Data_Value	Valore del dato (punteggio stimato)
Data_Value_Alt	Uguale al valore del dato, ma la formattazione è numerica
Data_Value_Footnote_Symbol	Simbolo da utilizzare per segnalare le note a piè di pagina
Data_Value_Footnote	Descrizione della nota a piè di pagina
LowConfidenceLimit	Limite inferiore dell'intervallo di confidenza sul dato al 95%
HighConfidenceLimit	Limite superiore dell'intervallo di confidenza sul dato al 95%
Break_Out_Category	Descrizione della categoria di ripartizione
Break_Out	Descrizione del gruppo di ripartizione
CategoryId	Valore di ricerca della categoria
TopicId	Valore di ricerca dell'oggetto della visita
IndicatorId	Valore di ricerca dell'indicatore della visita
Data_Value_TypeID	Valore di ricerca del tipo di valore del dato
BreakOutCategoryId	Valore di ricerca della categoria di ripartizione
BreakOutId	Valore di ricerca del gruppo di ripartizione
LocationId	Valore di ricerca della località della visita
GeoLocation	Latitudine e longitudine da fornire per la formattazione della geo-localizzazione o del geocodice nel formato (latitudine, longitudine)

	Year	LocationAbbr	LocationDesc	DataSource	PriorityArea1	PriorityArea2	PriorityArea3	PriorityArea4	Category	Topic	...	Break_Out_Category	Break_Out	CategoryId	TopicId
0	2006	US	United States	Medicare	NaN	NaN	NaN	NaN	Cardiovascular Diseases	Heart Failure	...	Race	Other	C1	T5
1	2005	US	United States	Medicare	NaN	NaN	NaN	NaN	Cardiovascular Diseases	Heart Failure	...	Race	Other	C1	T5
2	2007	US	United States	Medicare	NaN	NaN	NaN	NaN	Cardiovascular Diseases	Coronary Heart Disease	...	Age	65+	C1	T4
3	2008	US	United States	Medicare	NaN	NaN	NaN	NaN	Cardiovascular Diseases	Coronary Heart Disease	...	Gender	Female	C1	T4
4	2004	US	United States	Medicare	NaN	NaN	NaN	NaN	Cardiovascular Diseases	Heart Failure	...	Overall	Overall	C1	T5
...
42635	2012	CO	Colorado	Medicare	Million Hearts	NaN	NaN	NaN	Cardiovascular Diseases	Stroke	...	Gender	Male	C1	T6
42636	2010	MI	Michigan	Medicare	Million Hearts	NaN	NaN	NaN	Cardiovascular Diseases	Stroke	...	Gender	Female	C1	T6
42637	2012	ME	Maine	Medicare	Million Hearts	NaN	NaN	NaN	Cardiovascular Diseases	Stroke	...	Race	Hispanic	C1	T6
42638	2013	ID	Idaho	Medicare	Million Hearts	NaN	NaN	NaN	Cardiovascular Diseases	Stroke	...	Race	Non-Hispanic White	C1	T6
42639	2012	IA	Iowa	Medicare	Million Hearts	NaN	NaN	NaN	Cardiovascular Diseases	Stroke	...	Gender	Male	C1	T6

Figura 2.1: Alcuni dati presi dal dataset.

Il dataset ha una dimensione complessiva di 1236560 elementi. Nel dataset si alternano attributi di diverso tipo, tra cui *int64* e *float64*, ma si nota soprattutto una grande presenza di attributi di tipo *object*, che porterà alla conversione di essi in categorie per poter essere rappresentati.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42640 entries, 0 to 42639
Data columns (total 29 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Year              42640 non-null   int64  
 1   LocationAbbr      42640 non-null   object  
 2   LocationDesc       42640 non-null   object  
 3   DataSource         42640 non-null   object  
 4   PriorityArea1     9360 non-null    object  
 5   PriorityArea2     0 non-null       float64 
 6   PriorityArea3     14560 non-null   object  
 7   PriorityArea4     0 non-null       float64 
 8   Category          42640 non-null   object  
 9   Topic              42640 non-null   object  
 10  Indicator         42640 non-null   object  
 11  Data_Value_Type   42640 non-null   object  
 12  Data_Value_Unit   42640 non-null   object  
 13  Data_Value        42111 non-null   float64 
 14  Data_Value_Alt    42640 non-null   float64 
 15  Data_Value_Footnote_Symbol 529 non-null    object  
 16  Data_Value_Footnote 529 non-null    object  
 17  LowConfidenceLimit 42111 non-null   float64 
```

```
18 HighConfidenceLimit      42111 non-null float64
19 Break_Out_Category       42640 non-null object
20 Break_Out                42640 non-null object
21 CategoryId               42640 non-null object
22 TopicId                  42640 non-null object
23 IndicatorID              42640 non-null object
24 Data_Value_TypeID        42640 non-null object
25 BreakOutCategoryId       42640 non-null object
26 BreakOutId                42640 non-null object
27 LocationID               42640 non-null int64
28 GeoLocation              41820 non-null object
dtypes: float64(6), int64(2), object(21)
memory usage: 9.4+ MB
```

2.2 Valori 'null'

Abbiamo proceduto verificando il numero di valori mancanti per ogni attributo, con il seguente risultato:

Attributo	Numero di valori mancanti
Year	0
LocationAbbr	0
LocationDesc	0
DataSource	0
PriorityArea1	33280
PriorityArea2	42640
PriorityArea3	28080
PriorityArea4	42640
Category	0
Topic	0
Indicator	0
Data_Value_Type	0
Data_Value_Unit	0
Data_Value	529
Data_Value_Alt	0
Data_Value_Footnote_Symbol	42111
Data_Value_Footnote	42111
LowConfidenceLimit	529
HighConfidenceLimit	529
Break_Out_Category	0
Break_Out	0
CategoryId	0
TopicId	0
IndicatorID	0
Data_Value_TypeID	0
BreakOutCategoryId	0
BreakOutId	0
LocationID	0
GeoLocation	820

Come si può notare, alcuni attributi contengono molti valori 'null'. Per questo, abbiamo deciso di eliminare le colonne con più del 65% di valori mancanti. Quest'operazione si traduce in Python nel seguente modo:

```

threshold = 65/100 * df.shape[0]

for x in df.columns:
    if df[x].isna().sum() > threshold:
        df = df[df.columns.difference([x])]
```

Abbiamo scelto il 65% come *threshold* perchè in questo modo (nel nostro caso) riusciamo ad eliminare tutti gli attributi con più della metà dei valori 'null'. Inoltre, è bene usare come *threshold* un valore maggiore del 50-60% (dei valori 'null') quando si vuole eliminare un attributo, mentre per una frequenza di valori 'null' minore del 50% si possono imputare (*riempire*) i dati con valori standard (come la media o la moda della distribuzione dell'attributo) o con valori *outlier* (poco frequenti nella distribuzione dell'attributo).

Il risultato è un nuovo *Dataframe* contenente 23 colonne (non più 29), poichè le seguenti colonne sono state rimosse:

- *PriorityArea1*
- *PriorityArea2*
- *PriorityArea3*
- *PriorityArea4*
- *Data_Value_Footnote_Symbol*
- *Data_Value_Footnote*

2.3 Eliminazione degli attributi strettamente correlati

Una volta eliminati gli attributi con troppi valori 'null', siamo passati ad analizzare tutti gli attributi nel dettaglio. Prima ancora di creare i primi grafici per comprendere le eventuali correlazioni tra gli attributi, evidenziamo che ci sono alcuni attributi che già dalla descrizione del dataset fanno intendere la loro correlazione con altri.

Gli attributi che presentano questo problema con un altro attributo sono:

- *TopicId* con *Topic*
- *BreakOutCategoryId* con *Break_Out_Category*

- *CategoryId* con *Category*
- *LocationDesc* con *LocationAbbr*
- *LocationID* con *LocationAbbr*
- *GeoLocation* con *LocationAbbr*
- *Data_Value_TypeID* con *Data_Value_Type*
- *IndicatorID* con *Indicator*

Inoltre lo stesso attributo *Indicator* è fortemente correlato alla *class label*, il quale dovrebbe essere calcolato solo in combinazione con essa, quindi è un attributo "non disponibile" per nuovi dati. Il suo utilizzo nel training set porta ad avere una *data leakage* che si verifica quando i dati di addestramento contengono informazioni sull'obiettivo, ma dati simili non saranno disponibili quando il modello viene utilizzato per la predizione. Questo porta a prestazioni elevate sul *test set* (e forse anche sul *validation set*), ma il modello avrà scarse prestazioni in produzione.

Pertanto si è deciso di eliminare tutti questi attributi strettamente correlati assieme a *Indicator* per evitare questo genere di problema.

2.4 Conversione degli attributi di tipo object in categorici

Al fine di avere un'interfaccia conforme al dominio ed alle librerie utilizzate, nell'oggetto **pandas.DataFrame** è stato cambiato il tipo di dati *object* in *category*.

In particolare, gli attributi modificati sono:

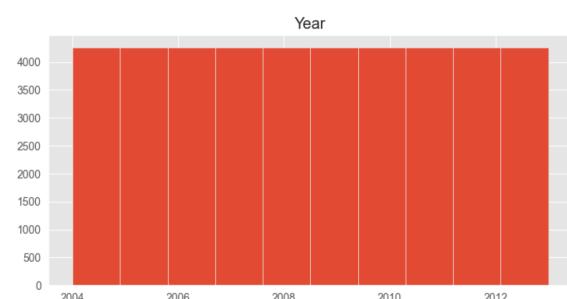
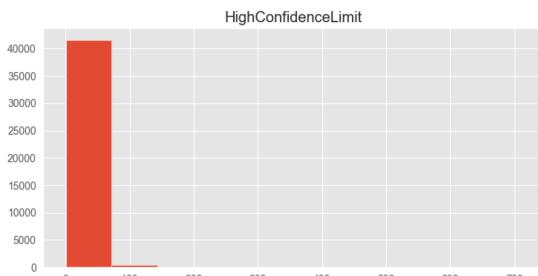
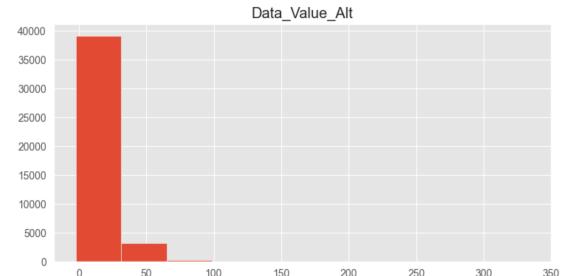
- Break_Out
- Break_Out_Category
- Category
- DataSource
- Data_Value_Type
- Data_Value_Unit
- LocationAbbr
- Topic

In questo modo gli attributi del Data Set diventano:

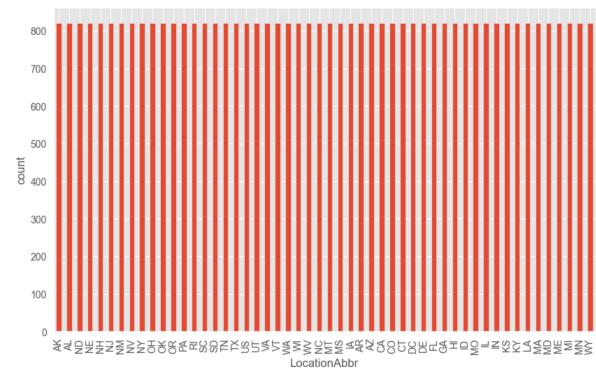
Attributo	Tipo
Break_Out	category
Break_Out_Category	category
Data_Value_Alt	float64
Data_Value_Unit	category
HighConfidenceLimit	float64
LocationAbbr	category
LowConfidenceLimit	float64
Topic	category

2.5 Istogrammi e grafici

Successivamente sono stati generati i grafici e gli istogrammi per ogni attributo. Di seguito sono riportati alcuni esempi:



((a)) Year



((b)) LocationAbbr

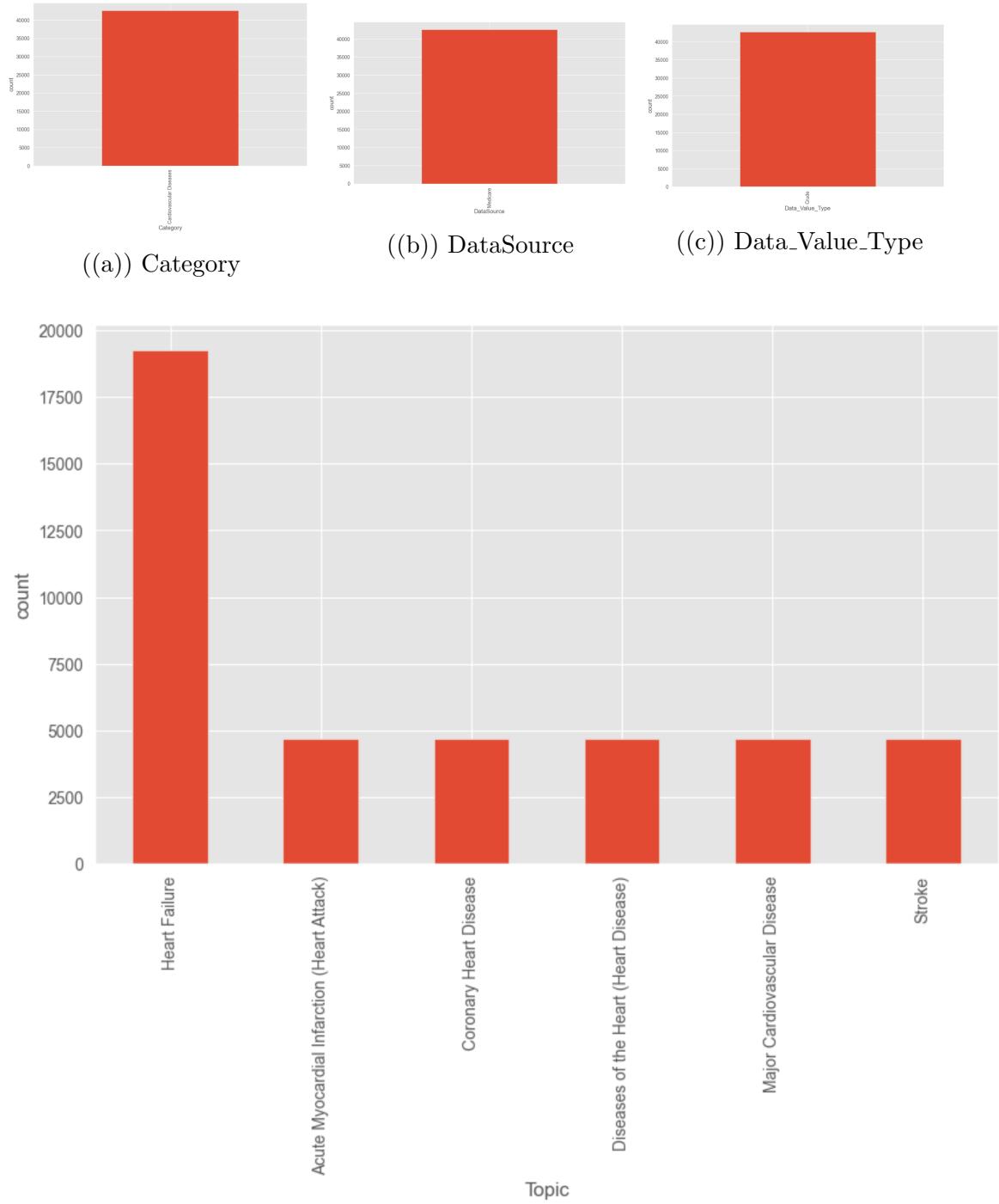
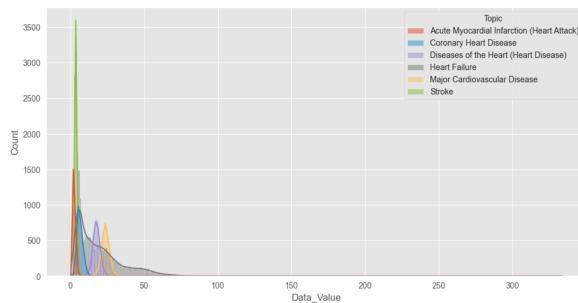
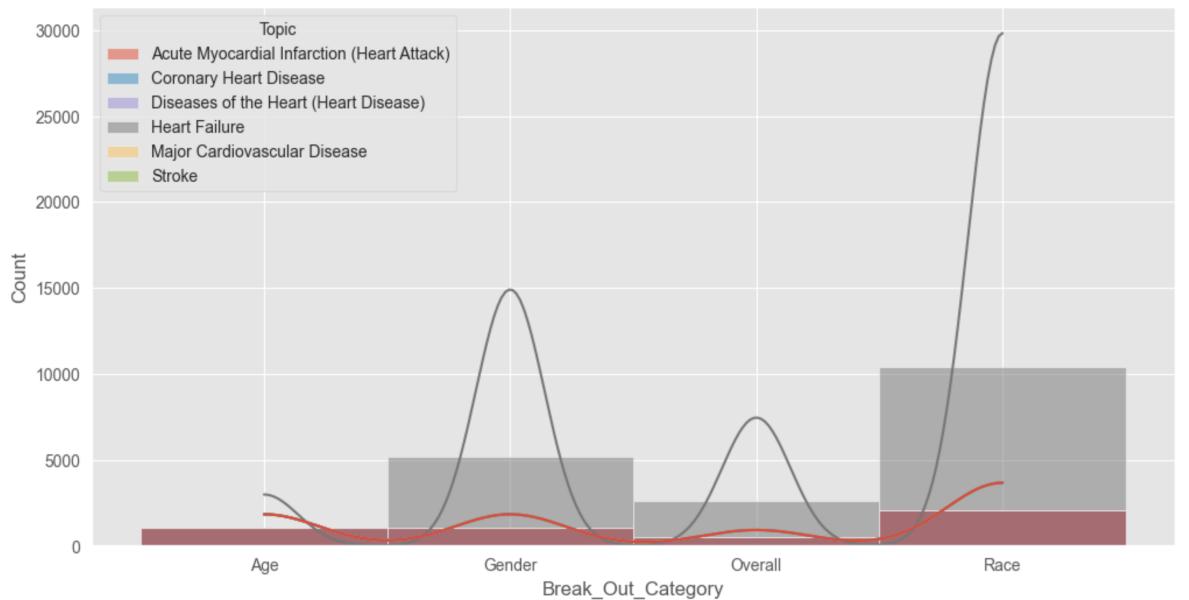


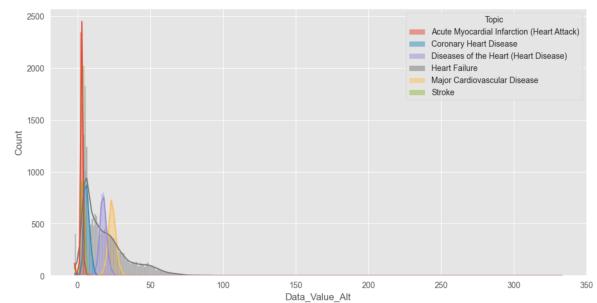
Figura 2.4: Istogramma riguardante la *class label Topic*.

Come si può notare, alcuni attributi hanno un solo valore (è il caso di Category, DataSource e Data_Value_Type), mentre altri hanno distribuzione uniforme (è il caso di Year e LocationAbbr). Inoltre, gli attributi Data_Value e Data_Value_Alt hanno distribuzione quasi identica.

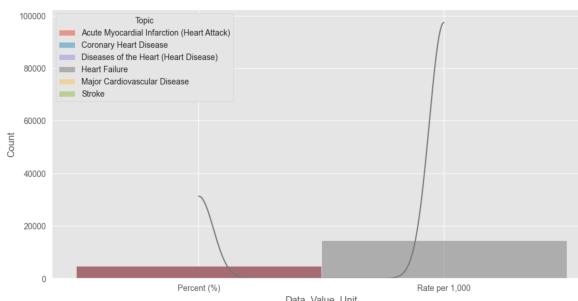
Di seguito, sono riportati i grafici relativi alle distribuzioni dei singoli attributi in correlazione con la variabile target *Topic*.



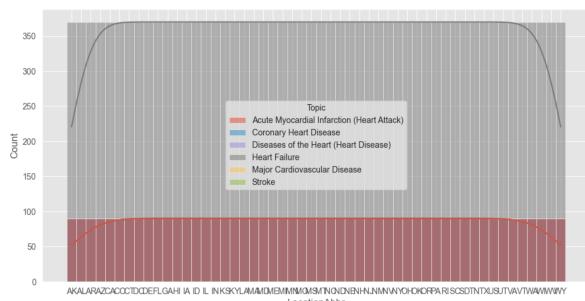
((a)) Data_Value



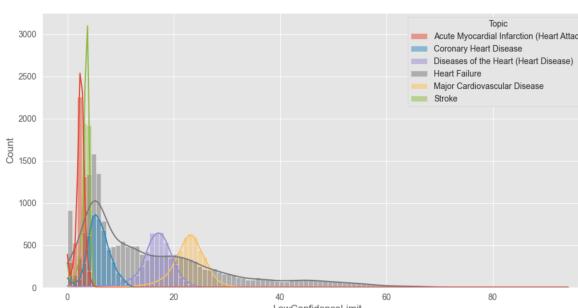
((b)) Data_Value_Alt



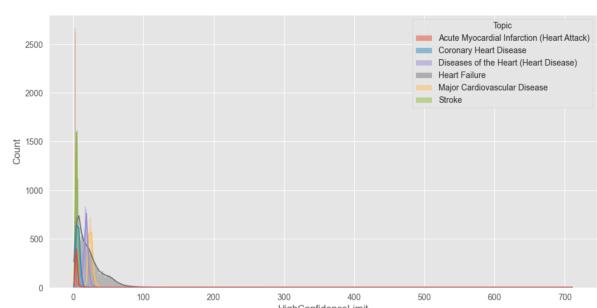
((a)) Data_Value_Unit



((b)) LocationAbbr



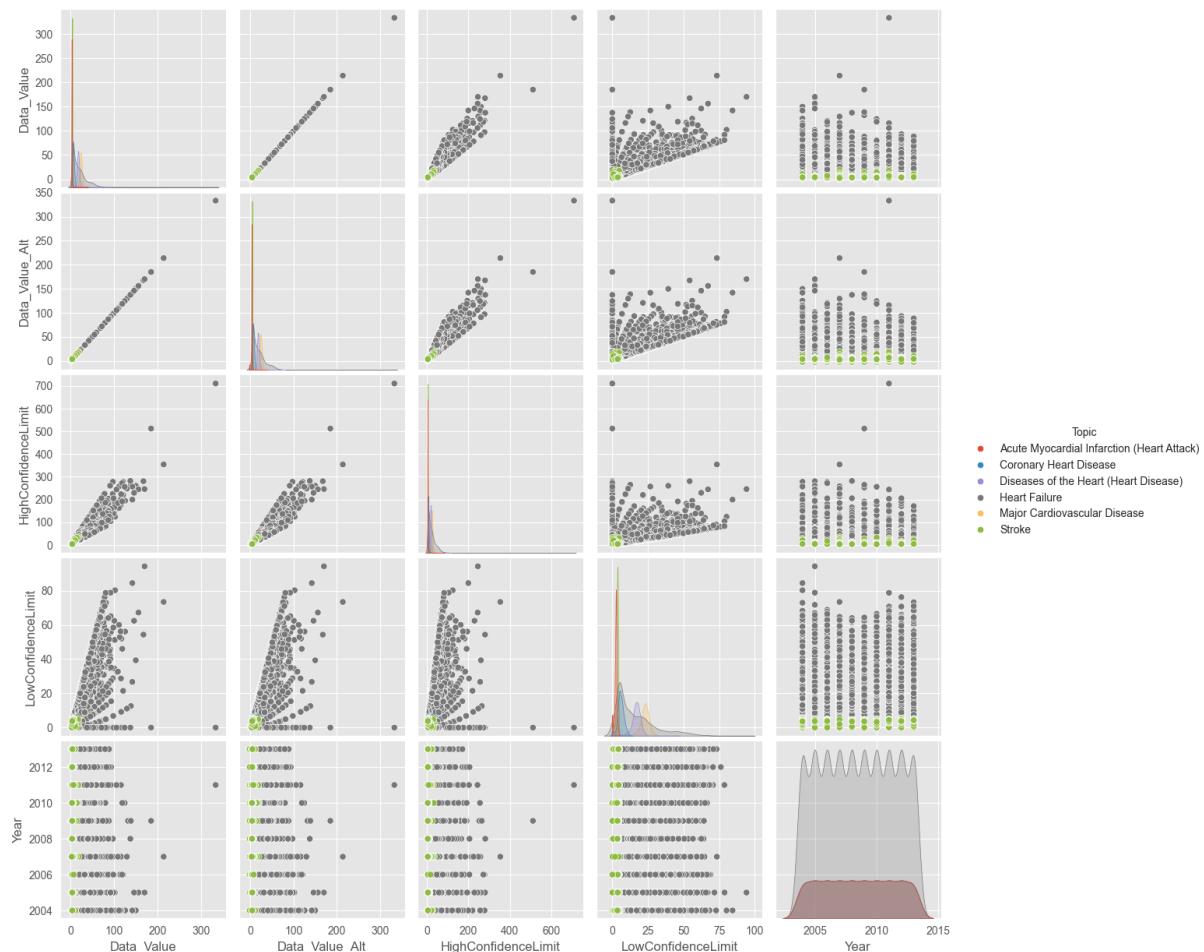
((a)) LowConfidenceLimit

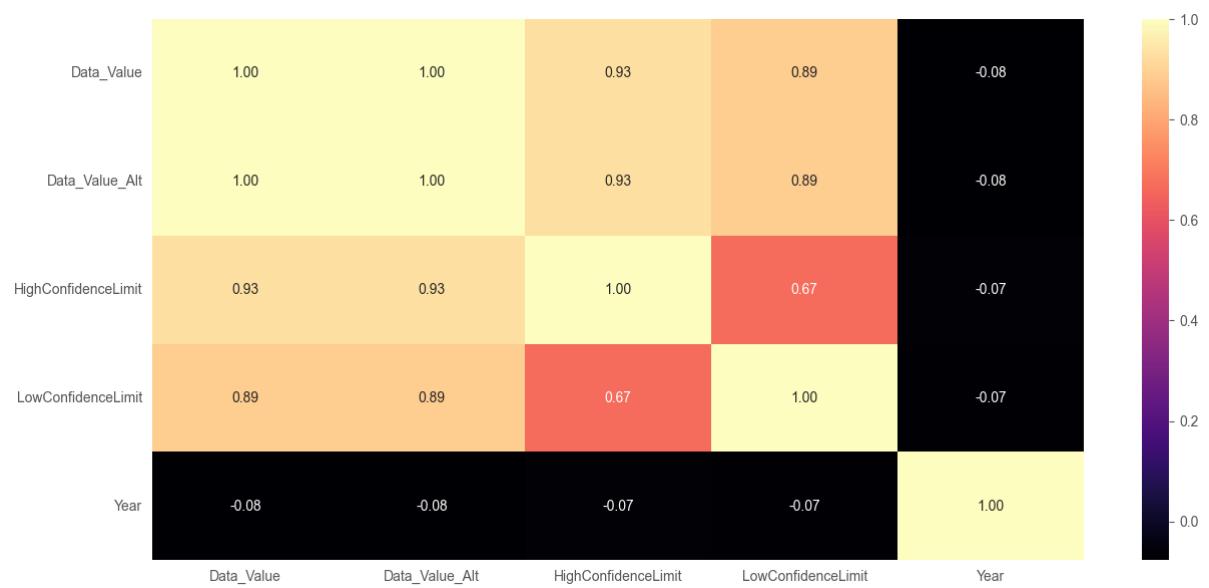


((b)) HighConfidenceLimit

La maggior parte dei grafici presenta una distribuzione sbilanciata, dovuta principalmente alla distribuzione eterogenea della variabile target *Topic*, come si può notare dal grafico in Figura 2.4 (c'è un'alta frequenza di valori *Heart Failure* rispetto a tutti gli altri).

Inoltre, è stato generato un **Pairplot** per visualizzare le relazioni tra le coppie di attributi e una **Heatmap** per verificare l'indice di correlazione tra attributi. Da quest'ultima, è facile intravedere ancora una volta una forte somiglianza tra gli attributi *Data_Value* e *Data_Value_Alt*, mentre si nota una distribuzione complementare tra *LowConfidenceLimit* e *HighConfidenceLimit*. Da ciò, si può intuire che *HighConfidenceLimit* e *LowConfidenceLimit* siano valori duali (riguardano lo stesso concetto, ma sono valori complementari).





Capitolo 3

Data preparation

3.1 Eliminazione degli ultimi attributi

Prima di passare alla fase di modellazione, si è notato attraverso i grafici come ci siano ancora alcuni attributi che possono essere eliminati poichè danno un contributo minore da diversi punti di vista rispetto ad altri.

Nello specifico, gli ultimi attributi che siamo andati a togliere sono:

- *Data_Value_Type*
- *Category*
- *DataSource*
- *Year*
- *Data_Value*

Category, *Data_Value_Type* e *DataSource* sono stati rimossi per via del loro scarso contributo al dataset, infatti tutti gli attributi in questione hanno al loro interno un unico valore, il quale in fase di modellazione non avrà molto rilievo poichè dalla sua prospettiva tutti gli elementi saranno uguali.

Per quanto riguarda *Data_Value* si era notato già dalla prima fase di comprensione del dominio (confermato in seguito dai grafici e dalla heatmap) quanto esso fosse molto simile a *Data_Value_Alt*. Abbiamo scelto di rimuovere *Data_Value* poichè al suo interno aveva qualche valore 'null', a differenza dell'altro attributo.

Infine, l'attributo *Year* è stato rimosso a causa della sua distribuzione uniforme notata dai grafici. Questo attributo non è l'unico che presenta questo problema, infatti un ulteriore attributo, *LocalizationAbbr*, presenta una distribuzione uniforme.

Si è scelto di mantenere l'attributo *LocalizationAbbr* poichè, una volta binarizzato, esso comporta un buon aumento delle prestazioni in tutti gli algoritmi, facendo capire come

sia molto utile all'interno del dataset, al contrario dell'attributo *Year*, che va addirittura a peggiorare le prestazioni in alcuni casi, soprattutto in quello che sarà evidenziato come **best model** più avanti.

3.2 Riempimento dei valori 'null'

Dopo aver rimosso gli attributi con una percentuale di valori 'null' superiore al 65%, nel Data Set sono rimasti gli attributi che presentano tali valori in una percentuale inferiore. In questo caso tali attributi sono:

- HighConfidenceLimit
- LowConfidenceLimit

Esistono diverse strategie per il "riempimento" dei valori 'null', per questo progetto sono state prese in considerazione le seguenti:

- **mode**, consiste nel sostituire i valori 'null' con la moda dei valori 'non-null';
- **mean**, consiste nel sostituire i valori 'null' con la media dei valori 'non-null'(non attuabile se il Data Set è estremamente sbilanciato causa l'inserimento di dati non reali);
- **outlier**, consiste nel sostituire i valori 'null' con valori conformi al tipo dell'attributo (per esempio numeri o stringhe) ma che sono "anomali" per l'attributo stesso.

Ognuna delle strategie sopra elencate è stata testata attraverso i modelli che verranno descritti nel capitolo successivo, ma i risultati empirici (fortemente influenzati dal numero di osservazioni del Data Set e dai modelli utilizzati) di ciascuna di esse sembrano convergere tutti verso gli stessi valori delle metriche analizzate.

E' stato dunque deciso di adottare la strategia **mode** in modo da assecondare la distribuzione dei valori del Data Set, inserendo il valore più frequente al posto dei valori 'null'.

Il codice python per implementare le strategie sopra descritte per gli attributi *HighConfidenceLimit* e *LowConfidenceLimit* è il seguente:

- **mode:**

```
def find_mode(vals) -> int:
    vals_count = {}

    # count the number of occurrences for each value
    for val in vals:
        if not val in vals_count.keys():
            vals_count[val] = 1
        else:
            vals_count[val] += 1

    # find the value with the maximum number of occurrences
    (mode, mode_count) = (-1, -1)
    for key in vals_count.keys():
        if vals_count[key] > mode_count:
            mode = key
            mode_count = vals_count[key]

    return mode

hcl_vals = df['HighConfidenceLimit'].dropna()
hcl_mode = find_mode(hcl_vals)

lcl_vals = df['LowConfidenceLimit'].dropna()
lcl_mode = find_mode(lcl_vals)

df.fillna(value={
    'HighConfidenceLimit' : hcl_mode,
    'LowConfidenceLimit' : lcl_mode
}, inplace=True)
```

- **mean:**

```

hcl_vals = df['HighConfidenceLimit'].dropna()
hcl_mean = sum(hcl_vals)/len(hcl_vals)

lcl_vals = df['LowConfidenceLimit'].dropna()
lcl_mean = sum(lcl_vals)/len(lcl_vals)

df.fillna(value={
    'HighConfidenceLimit' : hcl_mean,
    'LowConfidenceLimit' : lcl_mean
}, inplace=True)

```

- **outlier:**

```

df.fillna(value={
    'HighConfidenceLimit' : -1,
    'LowConfidenceLimit' : -1}, inplace=True)

```

3.3 Binarizzazione degli attributi

Prima di passare alla fase successiva sono stati trasformati i dati che normalmente gli algoritmi non accettano, ovvero le colonne con attributi non numerici, come *LocationAbbr*. In questo caso, si è applicata la binarizzazione di questi attributi per trasformarli in uno o più attributi binari.

Gli attributi che hanno avuto bisogno di questo processo prima di essere usati dagli algoritmi sono stati:

- *Break_Out*
- *Break_Out_Category*
- *Data_Value_Unit*
- *LocationAbbr*

Si nota inoltre che la binarizzazione dell'attributo *LocationAbbr* non è un processo semplice poichè genera molti nuovi attributi, uno per ogni località presente nel dataset. Tuttavia abbiamo notato che, includendolo, le prestazioni aumentano in maniera positiva nonostante abbia una distruzione uniforme; ciò non avviene per l'attributo *Year*, il quale, pur avendo una simile distribuzione uniforme, peggiora la prestazione di molti modelli, tra cui il nostro **best model**.

	Data_Value_Unit_Percent (%)	Data_Value_Unit_Rate per 1,000
0	True	False
1	True	False
2	True	False
3	True	False
4	True	False

Figura 3.1: Effetto della binarizzazione di *Data_Value_Type*.

	Break_Out_Category_Age	Break_Out_Category_Gender	Break_Out_Category_Overall	Break_Out_Category_Race
0	False	False	False	True
1	False	False	False	True
2	True	False	False	False
3	False	True	False	False
4	False	False	True	False

Figura 3.2: Effetto della binarizzazione di *Break_Out_Category*.

Esistono anche altre tecniche per trasformare gli attributi, come la discretizzazione, che nel nostro caso si poteva applicare agli attributi di tipo *float64* per dividerli in intervalli, ma abbiamo preferito non farlo perchè hanno distribuzioni molto differenti in relazione alla *class label*.

Capitolo 4

Modeling

4.1 Suddivisione del Data Set

Prima di generare i modelli di machine learning veri e propri, è necessario dividere il Data Set ottenuto dalle fasi precedenti in 3 insiemi disgiunti:

- **training set**, che servirà per addestrare i modelli;
- **validation set**, che verrà usato per valutare il modello (con feedback);
- **test set**, che verrà usato per valutare il modello (senza feedback).

Queste tre sezioni sono state distribuite, da traccia, nel seguente modo: il test set ricopre il 20% del Data Set originale, del restante 80% il validation set ricopre il 20% ed il training set l'80%.

Per fare ciò è stata usata la funzione *train_test_split* del package **sklearn.model_selection** ed il package **numpy**.

L'operazione di divisione del Data Set è effettuata dal seguente codice python:

```
x = np.array(df2.values)
y = np.array(df['Topic'].values)

seed = 12062024
test_size = .2
val_size = .2

x_train, x_test, y_train, y_test = (
    train_test_split(x, y, test_size=test_size, random_state=seed))
x_train, x_val, y_train, y_val = (
    train_test_split(x_train, y_train, test_size=val_size, random_state=seed))
```

4.2 Modelli

Una volta effettuato lo *splitting* del Data Set, è stato possibile creare i diversi modelli di machine learning dai quali verrà successivamente scelto il *best model* in base alle loro performance sul validation set.

Per questo progetto sono state scelte le seguenti tipologie di modelli:

- Decision Tree Classifier (usando come criterio di branching *entropy* e *gini index*)
- Naive Bayes Classifier
- MultiLayer Perceptron Classifier
- Random Forest Classifier
- K-Nearest Neighbors Classifier

Tutti i modelli utilizzati sono stati importati dalle seguenti librerie di **sklearn**:

- **tree**, per importare *DecisionTreeClassifier*
- **naive_bayes**, per importare *GaussianNB*
- **neural_network**, per *MLPClassifier*
- **ensemble**, per *RandomForestClassifier*
- **neighbors**, per *KNeighborsClassifier*

Il codice python usato per effettuare *training* e *validation* su i primi 3 modelli sopra elencati è il seguente:

```
models = []
models.append(
    'C45',
    DecisionTreeClassifier(criterion='entropy', random_state=seed))
models.append(
    'CART',
    DecisionTreeClassifier(criterion='gini', random_state=seed))
models.append(
    'GaussianNaiveBayes',
    GaussianNB())
)
```

```

models.append(
    'NeuralNetwork',
    MLPClassifier(
        hidden_layer_sizes=(50, 10, ),
        max_iter=500,
        verbose=True,
        random_state=seed)
)

# train each model in turn

for name, model in models:
    model.fit(x_train, y_train)

```

Per quanto riguarda il modello KNeighborsClassifier è stato usato il seguente codice:

```

knn = KNeighborsClassifier()
knn.fit(x_train, y_train)

```

Infine, per il meta-modello RandomForestClassifier è stato usato il codice che segue:

```

rf = RandomForestClassifier(n_estimators=50, random_state=seed)
rf.fit(x_train, y_train)

```

Dopo aver addestrato tutti i modelli, abbiamo generato, per ciascuno di essi, un report delle prestazioni (sul validation set) tramite la funzione *classification_report* del package **sklearn.metrics**.

Di seguito l'elenco di tali report:

Classification metrics:

	precision	recall	f1-score	support
Acute Myocardial Infarction (Heart Attack)	0.81	0.85	0.83	750
Coronary Heart Disease	0.64	0.65	0.64	711
Diseases of the Heart (Heart Disease)	0.93	0.91	0.92	762
Heart Failure	0.91	0.91	0.91	3085
Major Cardiovascular Disease	0.93	0.94	0.94	780
Stroke	0.69	0.66	0.67	735
accuracy			0.85	6823
macro avg	0.82	0.82	0.82	6823
weighted avg	0.85	0.85	0.85	6823

Figura 4.1: DecisionTreeClassifier (entropy).

Classification metrics:

	precision	recall	f1-score	support
Acute Myocardial Infarction (Heart Attack)	0.80	0.86	0.83	750
Coronary Heart Disease	0.63	0.67	0.65	711
Diseases of the Heart (Heart Disease)	0.93	0.92	0.92	762
Heart Failure	0.91	0.90	0.91	3085
Major Cardiovascular Disease	0.94	0.93	0.93	780
Stroke	0.69	0.64	0.66	735
accuracy			0.85	6823
macro avg	0.82	0.82	0.82	6823
weighted avg	0.85	0.85	0.85	6823

Figura 4.2: DecisionTreeClassifier (gini).

Classification metrics:

	precision	recall	f1-score	support
Acute Myocardial Infarction (Heart Attack)	0.59	0.60	0.59	750
Coronary Heart Disease	0.49	0.72	0.58	711
Diseases of the Heart (Heart Disease)	0.78	0.80	0.79	762
Heart Failure	1.00	0.76	0.86	3085
Major Cardiovascular Disease	0.81	0.79	0.80	780
Stroke	0.38	0.59	0.47	735
accuracy			0.73	6823
macro avg	0.68	0.71	0.68	6823
weighted avg	0.79	0.73	0.75	6823

Figura 4.3: GaussianNB (Naive Bayes).

Classification metrics:

	precision	recall	f1-score	support
Acute Myocardial Infarction (Heart Attack)	0.80	0.90	0.84	750
Coronary Heart Disease	0.76	0.67	0.71	711
Diseases of the Heart (Heart Disease)	0.93	0.97	0.95	762
Heart Failure	0.91	0.93	0.92	3085
Major Cardiovascular Disease	0.98	0.92	0.95	780
Stroke	0.74	0.67	0.70	735
accuracy			0.88	6823
macro avg	0.85	0.84	0.85	6823
weighted avg	0.87	0.88	0.87	6823

Figura 4.4: MLPClassifier (Neural Network).

Classification metrics: KNN

	precision	recall	f1-score	support
Acute Myocardial Infarction (Heart Attack)	0.84	0.85	0.85	750
Coronary Heart Disease	0.72	0.63	0.67	711
Diseases of the Heart (Heart Disease)	0.93	0.90	0.92	762
Heart Failure	0.90	0.92	0.91	3085
Major Cardiovascular Disease	0.93	0.93	0.93	780
Stroke	0.70	0.72	0.71	735
accuracy			0.86	6823
macro avg	0.84	0.82	0.83	6823
weighted avg	0.86	0.86	0.86	6823

Figura 4.5: KNeighborsClassifier (KNN).

Classification metrics: RandomForest

	precision	recall	f1-score	support
Acute Myocardial Infarction (Heart Attack)	0.84	0.86	0.85	750
Coronary Heart Disease	0.69	0.66	0.68	711
Diseases of the Heart (Heart Disease)	0.94	0.92	0.93	762
Heart Failure	0.91	0.92	0.91	3085
Major Cardiovascular Disease	0.94	0.95	0.94	780
Stroke	0.69	0.69	0.69	735
accuracy			0.86	6823
macro avg	0.84	0.83	0.83	6823
weighted avg	0.86	0.86	0.86	6823

Figura 4.6: RandomForestClassifier.

Basandoci principalmente sulla metrica *f1-score* ed in generale anche sull'*accuracy*, abbiamo scelto come *best model* la **Neural Network**.

Capitolo 5

Best model evaluation

5.0.1 Valutazione del best model sul test set

Per valutare le performance del **best model** (che nel nostro caso, è la **Neural Network**), abbiamo eseguito il modello su un **test set**, ricavato partizionando i dati del *data set originale*. In questo modo, abbiamo ottenuto un nuovo dataset indipendente dal dataset usato nella fasi di training. Dato come criterio di valutazione l'**accuracy** del modello, è stato ottenuto un valore di **0.88**.

Inoltre, è stata generata una **matrice di confusione** per valutare la percentuale di errore delle predizioni per ogni valore della class label Topic, nel seguente modo:

```
def make_confusion_matrix(cf, categories='auto',
                         cbar=True, cmap='Blues', title=None):
    group_counts = [f'{value}\n' for value in cf.flatten()]

    box_labels = [f'{v1}'.strip() for v1 in group_counts]
    box_labels = np.asarray(box_labels).reshape(cf.shape[0], cf.shape[1])

    sb.heatmap(cf, annot=box_labels, fmt='',
               cmap=cmap, cbar=cbar,
               xticklabels=categories, yticklabels=categories)

    plt.ylabel('True label')
    plt.xlabel('Predicted label')

    if title:
        plt.title(title)
```

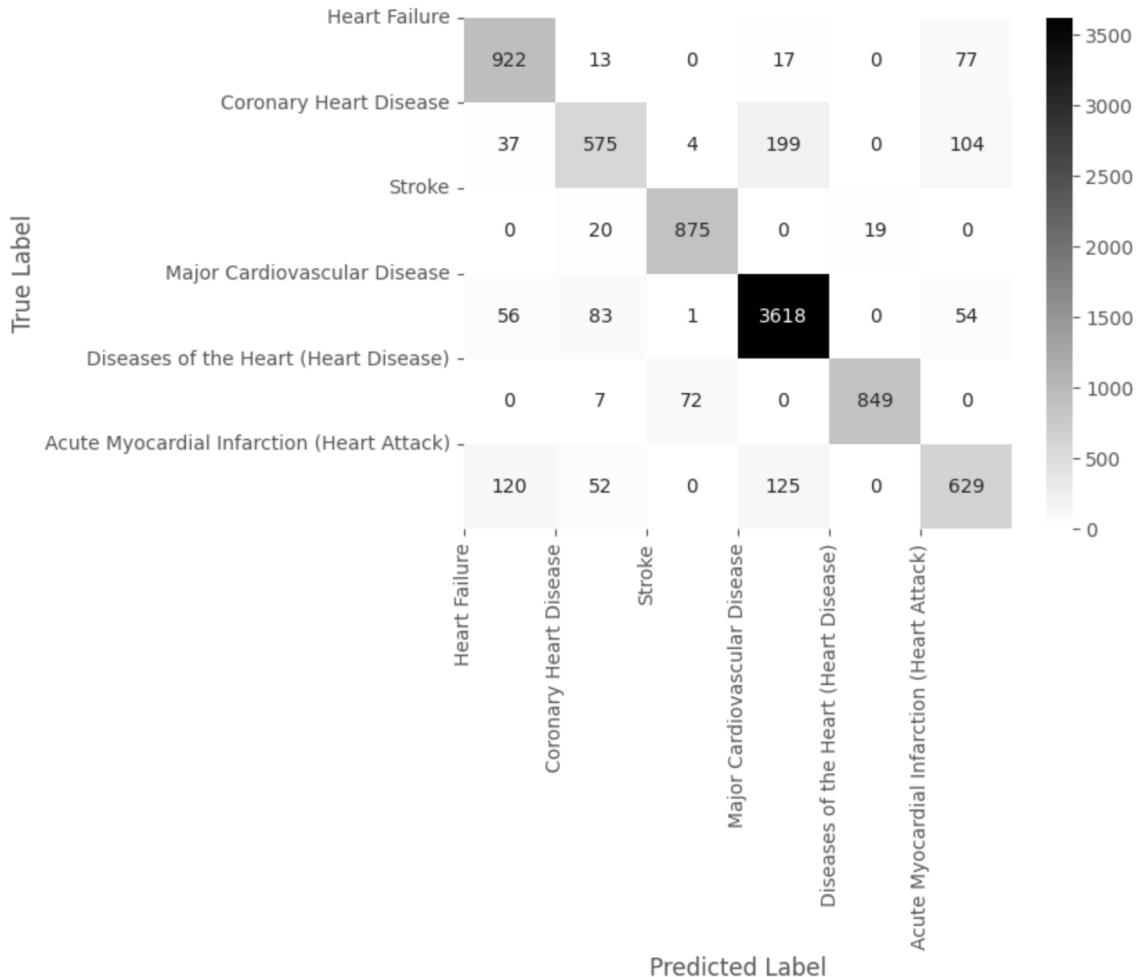
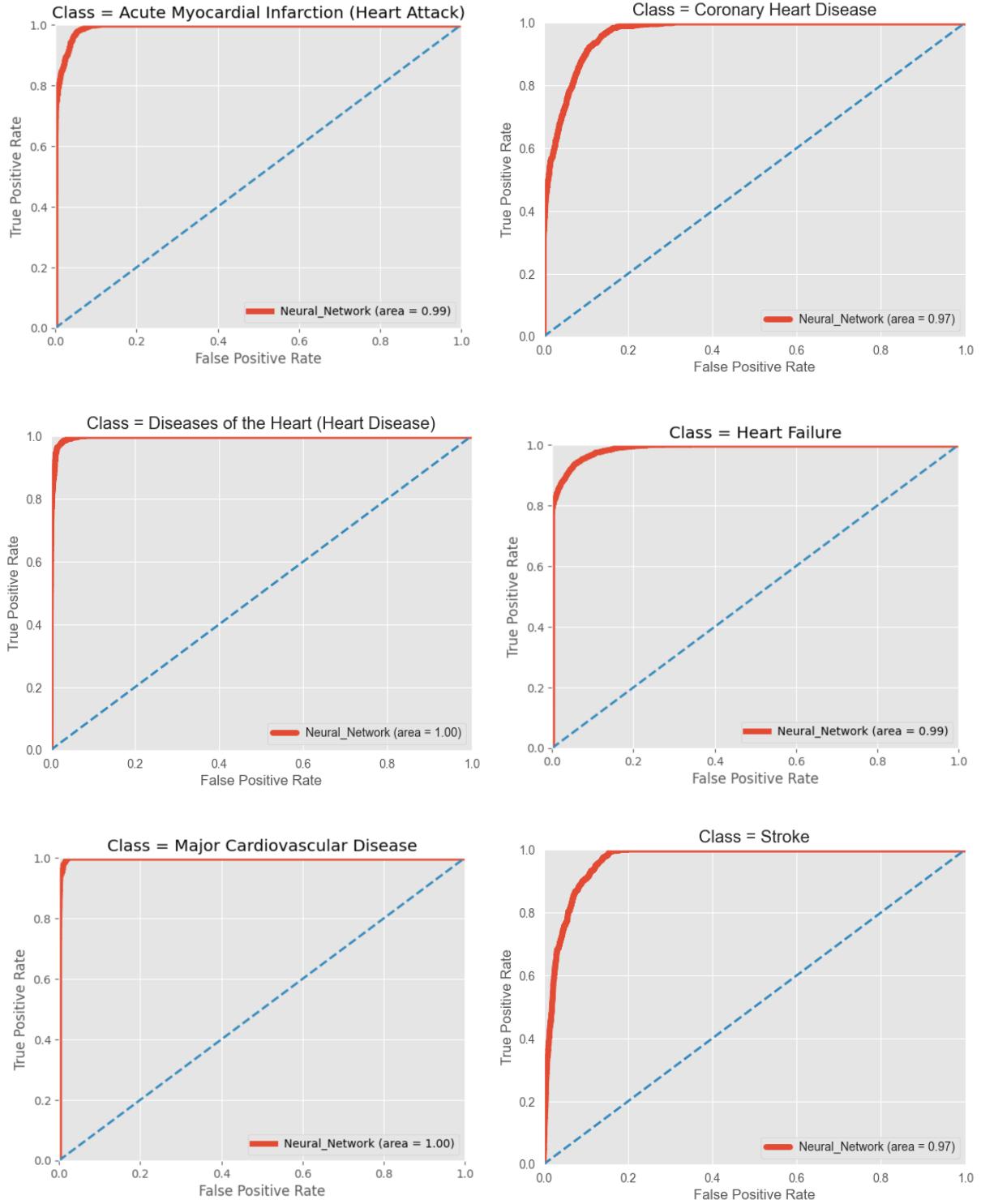


Figura 5.1: Matrice di confusione.

5.0.2 ROC curve

Infine, sono stati generati i grafici relativi alle **ROC curves** per ogni valore della class label Topic. Questi grafici permettono di valutare la percentuale di **false positive** dalle predizioni del modello (ovvero di istanze classificate erroneamente) rispetto alla percentuale di **true positive** (ovvero di istanze classificate correttamente).



Dalla matrice di confusione e dalle ROC curves, si evince una percentuale di errore alquanto bassa per tutti i valori di Topic. In particolare, si nota come il modello abbia una percentuale di errore tendente a zero per i valori *Disease of the heart (Heart Disease)* e *Major cardiovascular disease*: infatti, il valore della ROC curve (cioè **AUC** o *Area Under the Curve*, che si riferisce all'area sottesa alla curva) è pari a **1.00**, che indica una capacità di predizione quasi perfetta su questi valori (l'area sottesa alla curva è massima)

e un errore minimo per gli altri (*Acute myocardial infarction (Heart attack)* con AUC pari a **0.99**, *Coronary heart disease* con AUC pari a **0.97**, *Heart Failure* con **0.99** e *Stroke* con **0.97**). Inoltre, dai grafici possiamo intuire che il modello non sembra andare in **overfitting**, dato che la matrice di confusione e le ROC curves riportano le predizioni fatte sul test set: per andare in overfitting, un modello dovrebbe avere una capacità predittiva quasi perfetta sul training set ed essere poco efficace sul test set (e non è il nostro caso, avendo visto gli ottimi valori di AUC delle ROC curves).

Capitolo 6

Conclusioni

Dunque, le fasi svolte nel progetto sono state:

- **Data Understanding**, in cui sono stati visualizzati, tramite diagrammi e tabelle, i record presenti nel Data Set al fine di acquisire una migliore comprensione della natura stessa dei dati in oggetto;
- **Data Preparation**, in cui sono state apportate modifiche al Data Set in base alle informazioni raccolte nella fase precedente e alle nozioni teoriche studiate;
- **Modeling**, in cui sono stati generati diversi modelli di machine learning, i quali sono stati addestrati e validati attraverso il Data Set risultante dalle precedenti sezioni, e dai quali è stato scelto il *best model* in base ai valori di alcune metriche;
- **Evaluation**, in cui sono state valutate le performance del best model attraverso il test set; in questa sezione sono state analizzate anche la matrice di confusione e la ROC curve per avere una visione più accurata della capacità predittiva del *best model*.

In conclusione, dopo aver analizzato attentamente il dominio del Data Set siamo riusciti a generare un modello in grado di predire con sufficiente precisione, ma senza ricadere in *overfitting*, le malattie cardiache o ictus delle nuove osservazioni.