UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea Triennale in Informatica

SISTEMA INTELLIGENTE PER LA CLASSIFICAZIONE DELLA QUALITÁ DELL'ACQUA BASATO SU ALGORITMI DI APPREDIMENTO SUPERVISIONATO

–

GESTIONE DELLE DIVERSE TIPOLOGIE DI ACQUA ATTRAVERSO UNA KNOWLEDGE BASE

Github Repository: Giuseppecapozza/Progetto-ICON (github.com)

Autore: Capozza Giuseppe

Matricola: 708803

IDE UTILIZZATO: VsCode

LIBRERIE IMPORTATE:
- Numpy
- Pandas
- Matplotlib
- Seaborn
- Scikit-learn
- Pyswip
- Missingno

INTRODUZIONE

L'acqua potabile è importante per la salute pubblica, sia che venga utilizzata per bere, per uso domestico, per la produzione alimentare o per scopi ricreativi.
Acqua contaminata e scarse condizioni igienico-sanitarie sono legate alla trasmissione di malattie come colera, diarrea, dissenteria, epatite A, tifo e poliomielite. Servizi idrici e sanitari assenti, inadeguati o gestiti in modo inadeguato espongono le persone a rischi per la salute che sono invece prevenibili.
Quindi, ho preso ispirazione da questo per utilizzare questo set di dati sulla qualità dell'acqua per capire cosa costituisce un'acqua potabile sicura e applicare l'apprendimento automatico ad essa per distinguere tra acqua potabile e non potabile.

DATASET

Il dataset importato, presente nel repository GitHub, contiene varie sostanze chimiche che si ritrovano nell'acqua le quali stabiliscono la qualità dell'acqua rendendola o meno potabile:

1. ph: Acidità dell'acqua (da 0 a 14).

2. Hardness: Indica il contenuto di sali disciolti all'interno dell'acqua (in particolare calcio e magnesio) in mg/L.
Un'acqua troppo dura è responsabile della formazione di calcare.

3. Solids: Totale di solidi disciolti nell'acqua in ppm.

4. Chloramines: Quantità di clorammine nell'acqua in ppm.

5. Sulfate: Quantità di solfati disciolti nell'acqua in mg/L.

6. Conductivity: Conducibilità elettrica dell'acqua in µS/cm.

7. Organic_carbon: Quantità di carbonio organico nell'acqua in ppm.

8. Trihalomethanes: Quantità di tralometani nell'acqua in µg/L.

9. Turbidity: Misura della proprietà di emissione di luce dell'acqua in NTU.

10. Potability: Indica se l'acqua è sicura per il consumo umano. Potabile: 1 / Non potabile : 0.

Leggere e comprendere il dataset

Iniziamo con i seguenti passi:

1. Importare il dataset usando la libreria pandas.

2. Comprendere la struttura del dataset.

```
df = pd.read_csv('../ICON/Dataset/water_potability.csv')

df.head()
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

This DataSet Contains 3276 rows & 10 columns.

```
df.shape
```

```
(3276, 10)
```

```
df.describe()
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14.284970	66.396293	3.966786	0.390110
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	2.200000	0.738000	1.450000	0.000000
25%	6.093092	176.850538	15666.690297	6.127421	307.699498	365.734414	12.065801	55.844536	3.439711	0.000000
50%	7.036752	196.967627	20927.833607	7.130299	333.073546	421.884968	14.218338	66.622485	3.955028	0.000000
75%	8.062066	216.667456	27332.762127	8.114887	359.950170	481.792304	16.557652	77.337473	4.500320	1.000000
max	14.000000	323.124000	61227.196008	13.127000	481.030642	753.342620	28.300000	124.000000	6.739000	1.000000

```
df.info()
```

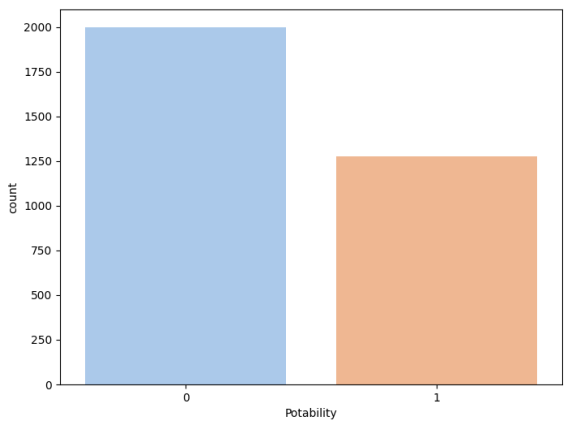
```
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   Potability       3276 non-null   int64  
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
```

Visualizzazione

```
pd.DataFrame(df['Potability'].value_counts())
```

	Potability
0	1998
1	1278

```
plt.figure(figsize=(8,6))
plt.tick_params(axis='x', which='major')
sb.countplot(x='Potability', data=df, palette='pastel')
plt.show()
```



Possiamo notare come il dataset non sia equilibrato.
Successivamente andremo a risolvere questa problematica per evitare pregiudizi nelle analisi successive.

```
fig = msno.matrix(df,color=(0,0.5,0.5))
```

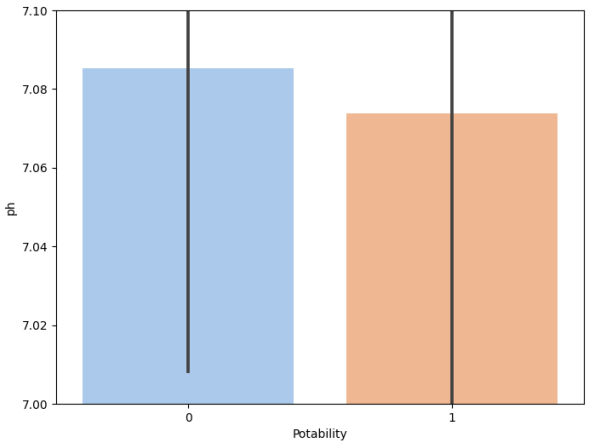
Da questo grafico possiamo notare come ci siano parecchi valori mancanti all'interno del dataset. Anche questa problematica verrà risolta in seguito.

Ora analizziamo i valori minini, massimi e medi di ogni caratteristica del dataset dividendoli secondo il valore della potabilità.

PH

```
max=pd.DataFrame(df.groupby("Potability")['ph'].max())
print(max)
mean=pd.DataFrame(df.groupby("Potability")['ph'].mean())
print(mean)
min=pd.DataFrame(df.groupby("Potability")['ph'].min())
print(min)
plt.figure(figsize=(8,6))
plt.tick_params(axis='both', which='major')
o=sb.barplot( x='Potability', y= 'ph',data=df, palette='pastel')
o.set_ylim(7,7.1)
```

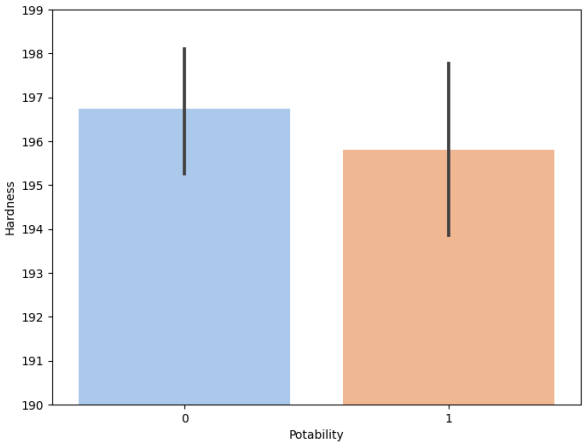
```
max
                  ph
Potability
0          14.000000
1          13.175402

mean
                  ph
Potability
0           7.085378
1           7.073783

min
                  ph
Potability
0           0.000000
1           0.227499
```



Il pH dell'acqua è una misura dell'equilibrio acido-base e, nella maggior parte delle acque naturali, è controllato dal sistema di equilibrio anidride carbonica-bicarbonato-carbonato. Un aumento della concentrazione di anidride carbonica abbasserà quindi il pH, mentre una diminuzione lo farà aumentare.
Il pH della maggior parte dell'acqua potabile è compreso tra 6,5 e 8,5.
Le acque naturali possono avere un pH più basso, a causa, ad esempio, di piogge acide o un pH più alto nelle aree calcaree.

DUREZZA

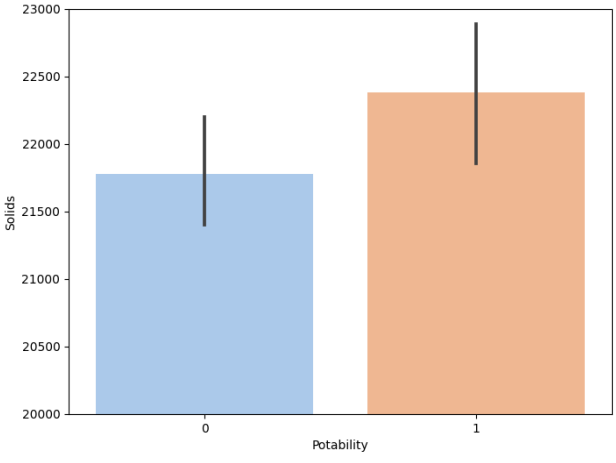
```
max
             Hardness
Potability
0          304.235912
1          323.124000

mean
             Hardness
Potability
0          196.733292
1          195.800744

min
             Hardness
Potability
0           98.452931
1           47.432000
```



La definizione di durezza dell'acqua è la quantità di calcio e magnesio disciolti nell'acqua. L'acqua dura è ricca di minerali disciolti, principalmente calcio e magnesio. Si potrebbe sentire gli effetti dell'acqua dura durante il lavaggio delle mani con del sapone. A seconda della durezza dell'acqua, dopo aver usato il sapone per lavare, si potrebbe sentire come se fosse rimasto un velo di residui sulle mani. Nell'acqua dura, il sapone reagisce con il calcio (che è relativamente alto nell'acqua dura) per formare "schiuma di sapone". Quando si utilizza acqua dura, è necessario più sapone o detersivo per pulire le cose, che si tratti di mani, capelli o bucato.

SOLIDI

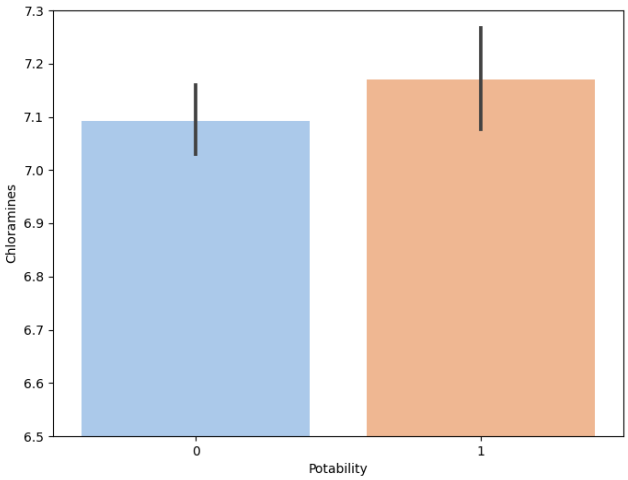
```
max
               Solids
Potability
0          61227.196008
1          56488.672413

mean
               Solids
Potability
0          21777.490788
1          22383.991018

min
               Solids
Potability
0          320.942611
1          728.750830
```



Con solidi si intende la concentrazione di particelle o solidi disciolti in acqua. Comprende sali inorganici come calcio, magnesio, cloruri, solfati, bicarbonati, ecc, insieme a molti altri composti inorganici che si dissolvono facilmente in acqua.

CLORAMMINE

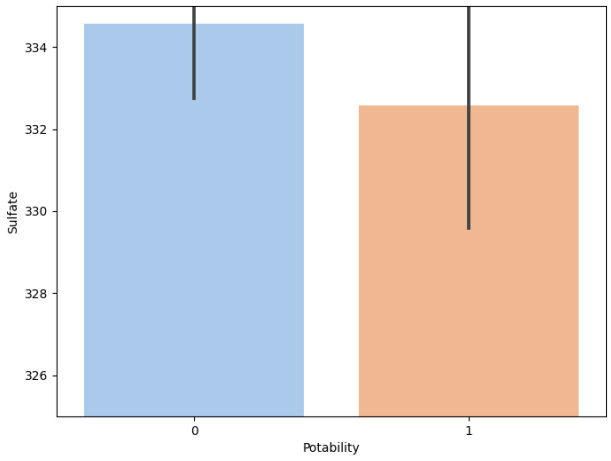
```
max
            Chloramines
Potability
0             12.653362
1             13.127000


mean
            Chloramines
Potability
0              7.092175
1              7.169338


min
            Chloramines
Potability
0              1.683993
1              0.352000
```



Le clorammine sono disinfettanti usati per il trattamento dell'acqua potabile e si formano più comunemente quando l'ammoniaca viene aggiunta al cloro per trattare l'acqua potabile.
Vengono utlizzati per fornire una disinfezione più duratura mentre l'acqua si sposta attraverso le tubazioni ai consumatori.
Le clorammine sono state utilizzate dai servizi idrici dagli anni '30.

SOLFATO

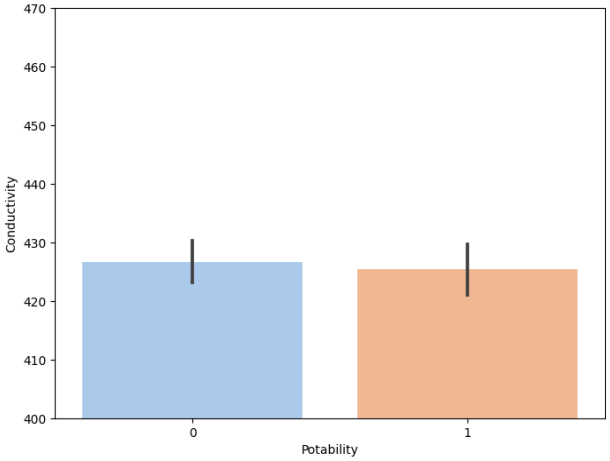
```
max
               Sulfate
Potability
0           460.107069
1           481.030642


mean
               Sulfate
Potability
0           334.56429
1           332.56699


min
               Sulfate
Potability
0           203.444521
1           129.000000
```



Il solfato (SO4) si trova in quasi tutte le acque naturali. L'origine della maggior parte dei composti solfati è l'ossidazione dei minerali solfiti, la presenza di scisti o rifiuti industriali.
Il solfato è uno dei principali componenti disciolti della pioggia. Alte concentrazioni di solfato nell'acqua che beviamo possono avere un effetto lassativo se combinate con calcio e magnesio, i due costituenti più comuni della durezza.

CONDUCIBILITÁ

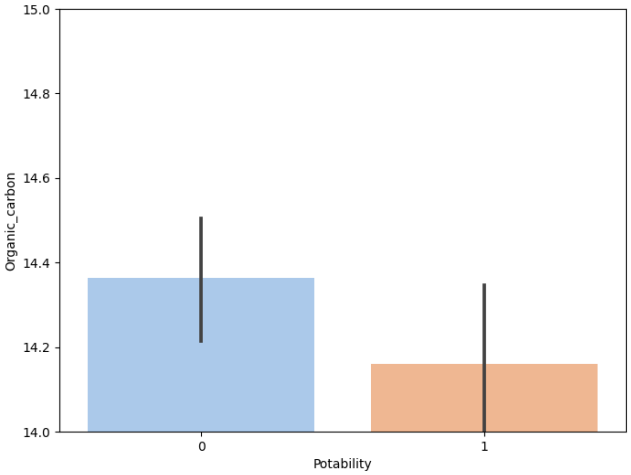
```
max
              Conductivity
Potability
0               753.342620
1               695.369528


mean
              Conductivity
Potability
0               426.730454
1               425.383800


min
              Conductivity
Potability
0               181.483754
1               201.619737
```



La conducibilità è una misura della capacità dell'acqua di far passare una corrente elettrica. Poiché i sali disciolti e altre sostanze chimiche inorganiche conducono corrente elettrica, la conduttività aumenta all'aumentare della salinità. La conducibilità è influenzata anche dalla temperatura: più calda è l'acqua, maggiore è la conducibilità.

CONTAMINANTI ORGANICI

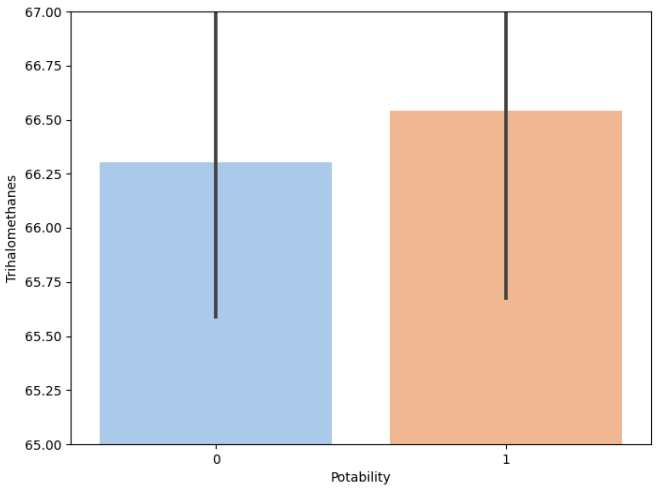
```
max
              Organic_carbon
Potability
0                  28.300000
1                  23.604298


mean
              Organic_carbon
Potability
0                  14.364335
1                  14.160893


min
              Organic_carbon
Potability
0                   4.371899
1                   2.200000
```



I contaminanti organici (sostanze organiche naturali, insetticidi, erbicidi e altri prodotti chimici agricoli) entrano nei corsi d'acqua durante il deflusso delle precipitazioni. Anche le acque reflue domestiche e industriali apportano contaminanti organici in varie quantità. A seguito di fuoriuscite o perdite accidentali, i rifiuti organici industriali possono entrare nei corsi d'acqua. Alcuni dei contaminanti potrebbero non essere completamente rimossi dai processi di trattamento; pertanto, potrebbero diventare un problema per le fonti di acqua potabile.
È importante conoscere il contenuto organico in un corso d'acqua.

TRIALOMETANI

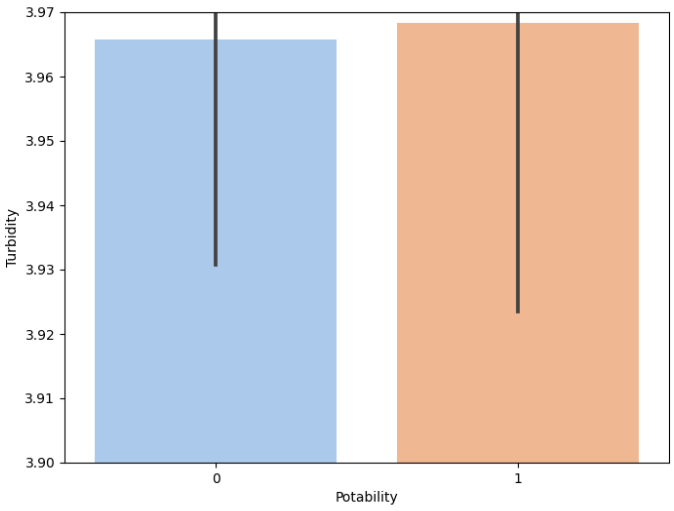
```
max
              Trihalomethanes
Potability
0                  120.030077
1                  124.000000

mean
              Trihalomethanes
Potability
0                   66.303555
1                   66.539684

min
              Trihalomethanes
Potability
0                    0.738000
1                    8.175876
```



I trialometani (THM) sono il risultato di una reazione tra il cloro utilizzato per la disinfezione dell'acqua del rubinetto e la materia organica naturale presente nell'acqua. A livelli elevati, i THM sono stati associati ad effetti negativi sulla salute come il cancro e gli esiti riproduttivi avversi.

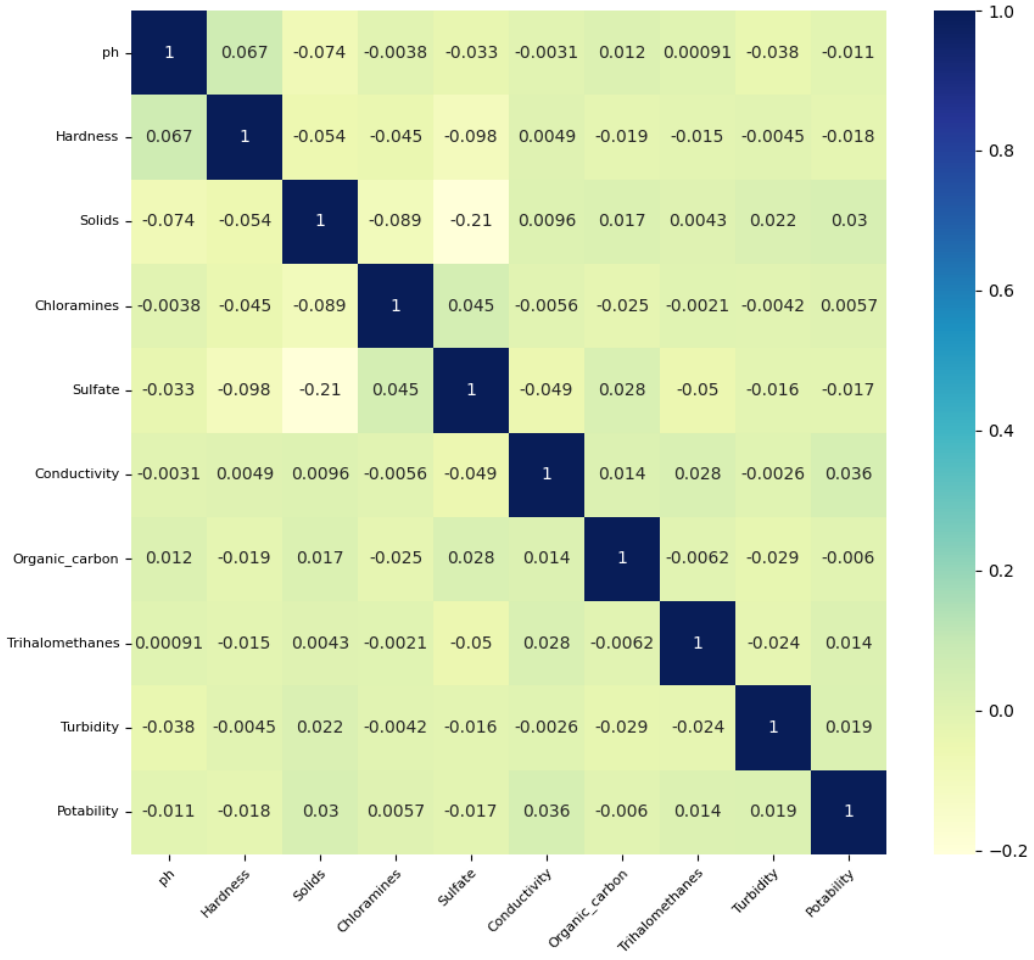
TORBIDITÁ

```
max
              Turbidity
Potability
0              6.739000
1              6.494249

mean
              Turbidity
Potability
0              3.965800
1              3.968328

min
              Turbidity
Potability
0              1.450000
1              1.492207
```



La torbidità è la misura della limpidezza relativa di un liquido. È una caratteristica ottica dell'acqua ed è una misura della quantità di luce che viene dispersa dal materiale nell'acqua quando una luce viene irradiata attraverso il campione d'acqua. Maggiore è l'intensità della luce diffusa, maggiore è la torbidità. Il materiale che rende l'acqua torbida include argilla, limo, materiale inorganico e organico molto piccolo, alghe, composti organici colorati disciolti e plancton e altri organismi microscopici.

HEATMAP



Come possiamo vedere, sembra esserci una correlazione molto bassa tra tutte le caratteristiche.

Ogni caratteristica, presa singolarmente, non risulta essere rilevante rispetto alla potabilità dell'acqua ma possiamo affermare che quest'ultima sia data dalla combinanzione di tutti questi valori.

Come abbiamo sottolineato prima, ci sono molti valori mancanti nel set di dati quindi cercherò di risolvere questo problema.

Prima di fare ciò, però, è necessario equilibrare il dataset; quindi vado a pareggiare i dati dell' acqua potabile eliminando tutti i dati in più riguardanti l'acqua non potabile.

```
zero  = df[df['Potability']==0]
one = df[df['Potability']==1]

resample = resample(zero, replace = True, n_samples = 1278)
df = pd.concat([one, resample])
df = shuffle(df)
```

Per risolvere il problema dei valori mancanti calcolo la media delle caratterische che necessitano quest' intervento in modo tale da riempire poi i valori mancanti con questi valori ottenuti.

```
df['ph'].fillna(value=df['ph'].median(),inplace=True)
df['Sulfate'].fillna(value=df['Sulfate'].median(),inplace=True)
df['Trihalomethanes'].fillna(value=df['Trihalomethanes'].median(),inplace=True)
```

STANDARDIZZAZIONE DEI DATI

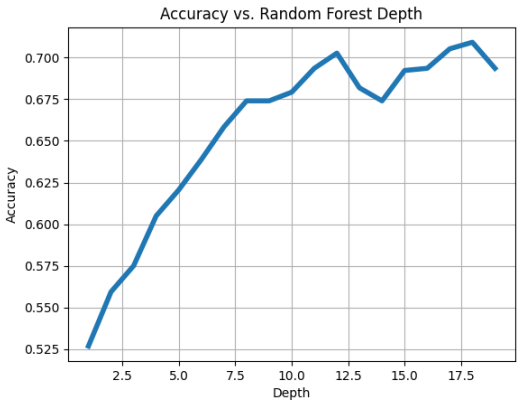
Procedo con la suddivisione in train e test e successivamente con la standardizzazione delle variabili numeriche.

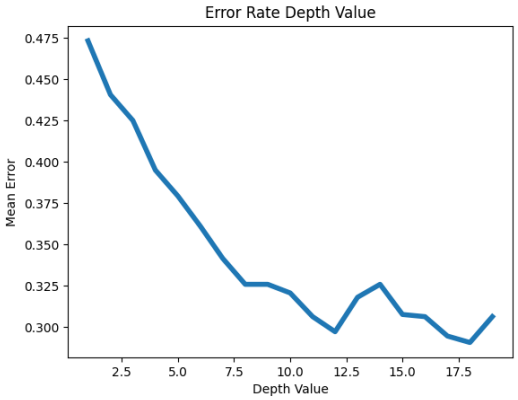
```
#train-test split
X = df.drop('Potability',axis=1).values
y = df['Potability'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

#standardizzazione
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

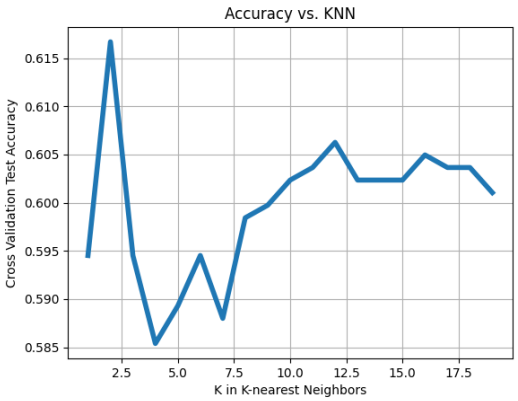
Dopo aver fatto ciò mi sono servito dei seguenti algoritmi: il KNN e il Random Forest. Innanzitutto, ho verificato quale fosse la profondità ideale del Random Forest e il valore più appropriato per l'iperparametro K del KNN che garantissero così il risultato più accurato.

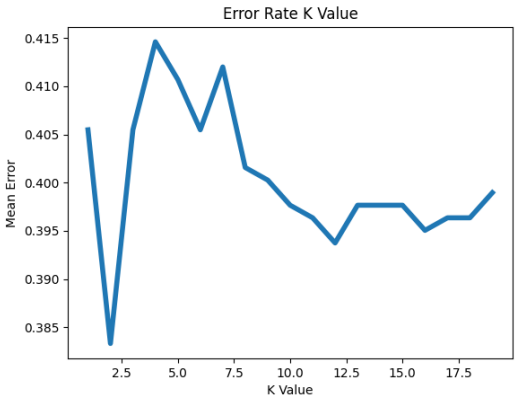
```
#RANDOM FOREST
accuracy = []
error = []
interval = np.arange(1, 20)
for i in interval:
    clf = RandomForestClassifier(max_depth=i, random_state=10)
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    accuracy.append(accuracy_score(y_test, predictions))
    error.append(np.mean(predictions!= y_test))
plt.plot(interval, accuracy, linewidth=4, markersize=10)
plt.title('Accuracy vs. Random Forest Depth')
plt.xlabel('Depth')
plt.ylabel('Accuracy')
plt.plot(interval,error,linewidth=4, markersize=10)
plt.title('Error Rate Depth Value')
plt.xlabel('Depth Value')
plt.ylabel('Mean Error')
#KNN
scores = []
error = []
for k in interval:
    knn = KNeighborsClassifier(p=k)
    knn.fit(X_train, y_train)
    predictions = knn.predict(X_test)
    scores.append(accuracy_score(y_test,predictions))
    error.append(np.mean(predictions!= y_test))
plt.plot(interval, scores, linewidth=4, markersize=10)
plt.title('Accuracy vs. KNN')
plt.xlabel("K in K-nearest Neighbors")
plt.ylabel("Cross Validation Test Accuracy")
```

```
plt.plot(interval,error,linewidth=4, markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```





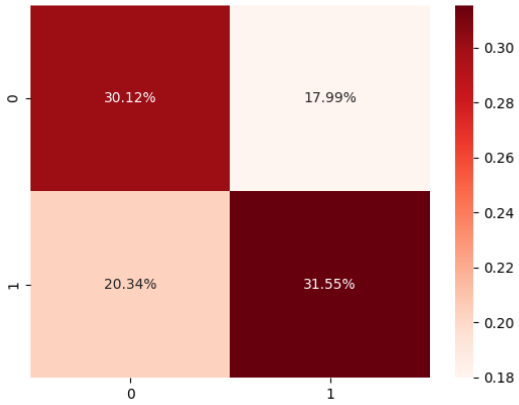




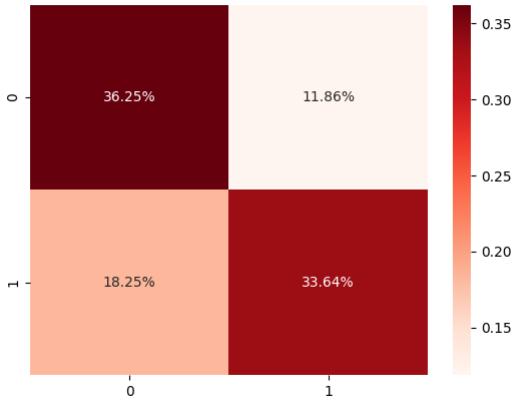
Dall'analisi delle prestazioni degli algoritmi emerge che l'algoritmo Random Forest ottiene risultati migliori con una profondità compresa tra 11 e 12, mentre il KNN con un valore K tra 1 e 2. Utilizzando questi valori ottengo quindi le seguenti prestazioni e le seguenti matrici di confusione, che mi confermano la maggior efficienza dell'algoritmo Random Forest:

```
#Creazione modello KNN
model_kn = KNeighborsClassifier(p=2)
#Allenamento
model_kn.fit(X_train, y_train)
#Predizione
pred_kn = model_kn.predict(X_test)
#Calcolo dell'accuratezza
kn = accuracy_score(y_test, pred_kn)
print(classification_report(y_test,pred_kn))
#confusion Maxtrix
cm5 = confusion_matrix(y_test, pred_kn)
sb.heatmap(cm5/np.sum(cm5), annot = True, fmt=  '0.2%', cmap = 'Reds')
#Creazione modello random forest
model_rf = RandomForestClassifier(max_depth=12, random_state=42)
#Allenamento
model_rf.fit(X_train, y_train)
#Predizione
pred_rf = model_rf.predict(X_test)
#Calcolo dell'accuratezza
rf = accuracy_score(y_test, pred_rf)
```

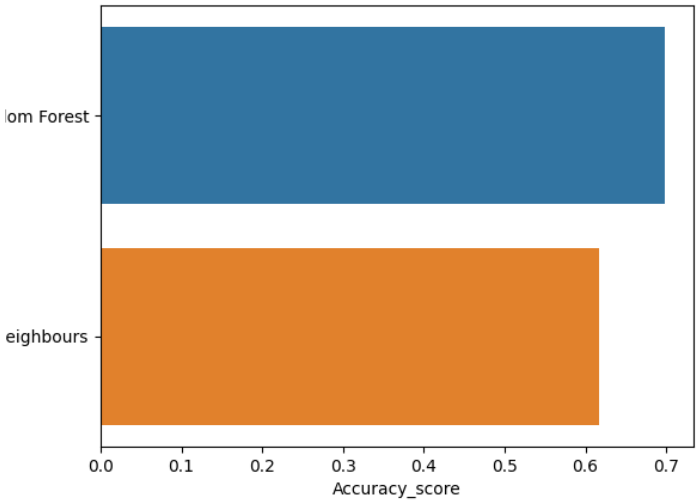
```
print(classification_report(y_test,pred_rf))
#confusion Maxtrix
cm3 = confusion_matrix(y_test, pred_rf)
sb.heatmap(cm3/np.sum(cm3), annot = True, fmt=  '0.2%', cmap = 'Reds')
```



KNN



RANDOM FOREST



PREDIZIONE POTABILITÁ ACQUA

```
df = pd.read_csv("../ICON/Dataset/Potabilityfinale.csv")
X = df.drop('Potability',axis=1).values
y = df['Potability'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

print('enter values')
ph = input("Enter ph value: ")
Hardness = input("Enter hardness value: ")
Solids = input("Enter solids value: ")
Chloramines = input("Enter chloramines value: ")
Sulfate = input("Enter sulfate value: ")
Conductivity = input("Enter conductivity value: ")
Organic_carbon = input("Enter organic carbon value: ")
Trihalomethanes = input("Enter trihalmethanes value: ")
```

```
Turbidity = input("Enter trurbidity value: ")

c = np.array([ph,Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic_carbon,
Trihalomethanes, Turbidity]).reshape(1,9)
forest = RandomForestClassifier(max_depth=12, random_state=42)
forest.fit(X_train, y_train)
forest_pred = forest.predict(c)
if forest_pred == 1:
    print("Potabile")
elif forest_pred == 0:
    print("Non potabile")
```

Verrà chiesto all'utente di inserire un valore per ogni caratteristica presente nel dataset e, grazie all'uso dell'algoritmo RandomForest, il quale abbiamo visto in precedenza avere maggior accuratezza nel risultato, avremo una predizione circa la potabilità o meno dell'acqua sulla base dei valori inseriti.

CREAZIONE KB

Ho successivamente creato una base di conoscenza da utilizzare per trovare il giusto tipo di acqua sulla base dell'età, del residuo o dei sintomi/condizioni fisiche che si posseggono.
Questa knowledge base è creata attraverso un algoritmo che legge i dati dell'acqua da un altro dataset, diverso da quello precedentemente utilizzato. Quest'ultimo dataset è stato creato da me raccogliendo informazioni su diversi tipi di acqua, la cui sorgente si trova nel nostro paese, cercando di coprire ogni possibile caratteristica.
Ho realizzato un algoritmo che legge le caratteristiche dell'acqua dal dataset estraendo quelle di interesse.
Si è quindi implementata una knowledge base nella quale sono stati inseriti:

-marca
-residuo
-bicarbonati
-solfato
-cloruri
-calcio
-magnesio
-fluoruro
-ph
-sodio

Di seguito parte del codice necessario per la creazione del kb:

```
dfKB = pd.read_csv('../ICON/Dataset/acquacsv.csv')

KNOWLEDGE_BASE = "../ICON/Dataset/potability.pl"
def is_empty(s):
    if s and s.strip():
        return True
    return False

file_data = ""
dfKB['marca'] = dfKB['marca'].astype(str)
dfKB['residuo'] = dfKB['residuo'].astype(str)
dfKB['bicarbonati'] = dfKB['bicarbonati'].astype(str)
dfKB['solfato'] = dfKB['solfato'].astype(str)
dfKB['cloruri'] = dfKB['cloruri'].astype(str)
```

```
dfKB['calcio'] = dfKB['calcio'].astype(str)
dfKB['magnesio'] = dfKB['magnesio'].astype(str)
dfKB['fluoruro'] = dfKB['fluoruro'].astype(str)
dfKB['ph'] = dfKB['ph'].astype(str)
dfKB['sodio'] = dfKB['sodio'].astype(str)


# Generating marca - residuo
for row in dfKB.itertuples():
    marca = row[1]
    residuo = row[2]
    string = "marca-residuo(\"" + marca + "\",\"" + residuo + "\")."

    if is_empty(marca) and is_empty(residuo) and (string not in file_data):
        file_data += string + "\n"

file_data += "\n"

# Generating marca - residuo - bicarbonati
for row in dfKB.itertuples():
    marca = row[1]
    residuo = row[2]
    bicarbonati = row[3]
    string = "marca-residuo-bicarbonati(\"" + marca + "\",\"" + residuo + "\",\"" +
bicarbonati + "\")."

    if is_empty(marca) and is_empty(residuo) and is_empty(bicarbonati) and (string not in
file_data):
        file_data += string + "\n"

file_data += "\n"
```

SCRITTURA KB

```
knowledge_base = open(KNOWLEDGE_BASE, mode="w")
knowledge_base.write(file_data)
knowledge_base.close()
print("\nFile created in: ", KNOWLEDGE_BASE)
```

FUNZIONI BASE DELLA KNOWLEDGE BASE

Si sono create funzioni di gestione di clausole e query.
Troviamo infatti funzioni di aggiunta e cancellazione di clausole e di interrogazione della knowledge base in Prolog:

```
prolog = Prolog()
prolog.consult("../ICON/Dataset/potability.pl")

def addAssert(prolog, str):
    prolog.assertz(str)
def deleteAssert(prolog, str):
    prolog.retract(str)
def query(prolog, str):
```

```
    qr = (str + ".")
    return list(prolog.query(qr))
```

Di seguito, come esempio, parte del codice utilizzato:

```
def menu():
    print("Benvenuto")
    answer = input("Scegliere parametro per la ricerca dell'acqua: \n"
                   "1) Residuo fisso \n"
                   "2) Eta \n"
                   "3) Sintomi/Condizioni fisiche \n"
                   "X) USCITA \n"
                   "")
    while answer[0] != ("x") and answer[0] != ("X"):
        if answer[0] == "1":
            residuo()
        elif answer[0] == "2":
            eta()
        elif answer[0] == "3":
            sintomi()
        else:
            print("RISPOSTA ERRATA!")
        answer = input("Scegliere parametro per la ricerca dell'acqua: \n"
                       "1) Residuo fisso \n"
                       "2) Eta \n"
                       "3) Sintomi/Condizioni fisiche \n"
                       "X) USCITA \n"
                       "")

def residuo():
    answer1 = input("Selezionare il tipo di acqua in base al residuo fisso:\n"
                "1) Alto (Uso terapeutico)(>1500 mg/L)\n"
                "2) Oligominerale (50-500 mg/L)\n"
                "3) Medio-minerale (500-1500 mg/L)\n"
                "4) Leggera (<50 mg/L)\n"
                "X) Torna al menu principale \n"
                )
    while answer1[0] != ("x") and answer1[0] != ("X"):
        if answer1[0] == "1":
            alto()
        elif answer1[0] == "2":
            oligominerale()
        elif answer1[0] == "3":
            mediominerale()
        elif answer1[0] == "4":
            leggera()
        else:
            print("RISPOSTA ERRATA!")
        answer1 = input("Selezionare il tipo di acqua in base al residuo fisso:\n"
                "1) Alto (Uso terapeutico)(>1500 mg/L)\n"
                "2) Oligominerale (50-500 mg/L)\n"
                "3) Medio-minerale (500-1500 mg/L)\n"
                "4) Leggera (<50 mg/L)\n"
```

```
                "X) Torna al menu principale \n"
                )
def alto():
    print("Acqua con alto residuo fisso (>1500 mg/L) \n")
    prez=3500
    prez = float(prez)
    interval = np.arange(1500,prez, 1)
    my_list = []
    for i in interval:
     i = i.astype(str)
     if (query(prolog, "marca-residuo(Marca,\"" + i + "\")")!= []):
        my_list.append(query(prolog, "marca-residuo(Marca,\"" + i + "\")"))
    print(my_list)
```

CONCLUSIONE

Uno dei possibili sviluppi futuri è quello di ampliare il dataset in modo tale da avere più scelte a disposizione per ogni richiesta di eventuali clienti o anche ampliare i sintomi che i tipi diversi di acqua mirano a curare/prevenire.
Oltre questo potrebbe essere opportuno sviluppare un'interfaccia software in modo tale da rendere l'esperienza utente decisamente più apprezzabile e usufruibile dopodiché far eseguire questo software su dei dispositivi, come ad esempio tablet, che verranno collocati all'interno di supermercati cosicché i clienti possano utilizzarli per cercare i tipi di acqua a loro utili sulla base delle marche presenti all'interno del supermercato.