Project 3: Water Quality Prediction using Machine Learning

Author: Nathalie Abell

Date: 06/02/2024

Install JAR packages:

```
!pip install -U -f https://pyspark.ai/ spark[ml]
!pwd
!wget -nc https://repo1.maven.org/maven2/org/xerial/snappy/snappy-0.2/snappy-0.2.jar
```

Importing the necessary libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import os, sys, gc, warnings, math, datetime
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier,
GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import auc
from sklearn.metrics import precision_recall_curve, average_precision_score
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import PrecisionRecallDisplay
%pip install -qq -U -f https://pyspark.ai/ spark[ml]
from pyspark.ml.classification import LinearSVC
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.classification import DecisionTree
from pyspark.ml.classification import RandomForest
from pyspark.ml.classification import GBTClassifier
from pyspark.ml.feature import StandardScaler
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StringIndexer
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.tuning import ParamGridBuilder
from pyspark.ml.tuning import CrossValidator
from pyspark.ml.tuning import TrainValidationSplit
from pyspark.ml import Pipeline
from pyspark.ml.pipelines import PipelineModel
from pyspark.ml.feature import MinMaxScaler
from pyspark.ml.feature import StandardScaler
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.feature import StringIndexerModel
from pyspark.ml.feature import VectorIndexerModel
from pyspark.ml.feature import VectorSlicer
from pyspark.ml.feature import Tokenizer
from pyspark.ml.feature import Word2Vec
```

Mounting Google Drive:

```
# Mounting Google Drive
from google.colab import drive
drive.mount('/content/drive')
ls -ltrh /content/drive/MyDrive/Colab\ Notebooks/Projects/*.csv
cp -v /content/drive/MyDrive/Colab\ Notebooks/Projects/*.csv .
ls -ltrh *.csv
ls -ltrh /content/drive/MyDrive/Colab\ Notebooks/Projects/
ls -ltrh *.csv
```

```
Mounted at /content/drive
ls: cannot access '/content/drive/MyDrive/Colab Notebooks/Projects/': No such file or directory
'/content/drive/MyDrive/Colab Notebooks/Projects/water_potability.csv' -> './water_potability.csv'
-rw-r--r-- 1 root root 355K Jan 30 19:51 water_potability.csv
ls: cannot access '*.csv': No such file or directory
```

Loading the dataset into a dataframe, displaying its size and head values:

```
df = pd.read_csv('water_potability.csv')
df.shape
df.head(5)
```

```
(3276, 10)
```

	ph	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability	Chlorine	Hardness	Solids
0	NaN	268.646941	564.308654	10.379783	86.990970	2.963135	0	7.300212	204.890455	20791.318981
1	3.716080	333.775777	592.885359	15.180013	56.329076	4.500656	0	NaN	129.422921	18630.057858
2	8.099124	NaN	418.606213	16.868637	66.420093	3.055934	0	6.635246	224.236259	19909.541732
3	8.316766	304.548754	475.492474	13.688024	100.341674	4.628771	0	9.077200	214.373394	22018.417441
4	9.092223	430.486851	389.375566	12.707760	31.997993	4.075075	0	1.571731	181.101509	17207.402449

5 rows × 10 columns

Data exploration

```
df.info()
```

Displaying a table representing the characteristics of the dataframe:

```
df.describe()
```

```
df.nunique()
```

Displaying an pie representing the number of unknowns in this object:

```
df.isnull().sum()
```

```
ph                 491
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes     162
Turbidity            0
Potability           0
Chlorine           158
Hardness             0
Solids               0
dtype: int64
```

Displaying a table of the characteristics of the dataframe when duplicates are deleted:

```
df.drop_duplicates().shape
```

```
(3276, 10)
```

Displaying the number of duplicates:

```
df.shape - df.drop_duplicates().shape
```

```
(0, 0)
```

Displaying a table representing the characteristics of the dataframe when duplicates are deleted:

```
df.drop_duplicates().describe()
```

	ph	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability	Chlorine	Hardness	Solids
count	2785.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000	2822.000000	3276.000000	3276.000000
mean	7.080795	333.775777	426.205111	14.284970	66.396293	3.949111	0.390110	7.234663	196.369496	22014.092526
std	1.594320	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849	1.594320	32.879761	8768.570828
min	0.000000	129.000000	181.483754	2.200000	8.577013	1.450000	0.000000	0.352000	47.432000	320.942611
25%	6.093092	307.699498	365.772042	12.065762	55.844536	3.439711	0.000000	6.127421	176.850538	15666.690297
50%	7.036752	333.073546	423.445884	14.218338	66.622485	3.955028	0.000000	7.130299	196.967627	20927.833607
75%	8.062316	359.948574	481.792304	16.557652	77.337473	4.500320	1.000000	8.114887	216.667456	27332.762127
max	14.000000	481.030642	753.342620	28.300763	124.000000	6.739191	1.000000	10.779375	323.124160	61227.196008

```
df.describe()
```

	ph	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability	Chlorine	Hardness	Solids
count	3276.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000	2822.000000	3276.000000	3276.000000
mean	7.080795	333.775777	426.205111	14.284970	66.396293	3.949111	0.390110	7.234663	196.369496	22014.092526
std	1.594320	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849	1.594320	32.879761	8768.570828
min	0.000000	129.000000	181.483754	2.200000	8.577013	1.450000	0.000000	0.352000	47.432000	320.942611
25%	6.093092	307.699498	365.772042	12.065762	55.844536	3.439711	0.000000	6.127421	176.850538	15666.690297
50%	7.036752	333.073546	423.445884	14.218338	66.622485	3.955028	0.000000	7.130299	196.967627	20927.833607
75%	8.062316	359.948574	481.792304	16.557652	77.337473	4.500320	1.000000	8.114887	216.667456	27332.762127
max	14.000000	481.030642	753.342620	28.300763	124.000000	6.739191	1.000000	10.779375	323.124160	61227.196008

5 rows × 10 columns

Data cleaning

Convert strings to lower or proper case:

```
df.columns = df.columns.str.lower()
```

Displaying the name, dtypes, function to clean columns names in dpm and:

```
def clean_cols(df):
    return df.rename(columns=lambda c: c.strip().lower().lower.replace(' ', '_').
              replace('(', '').replace(')', '').replace('?', '').
              replace('.', ''))
df = clean_cols(df)
df.dtypes
```

```
ph                 float64
sulfate            float64
conductivity       float64
organic_carbon     float64
trihalomethanes    float64
turbidity          float64
potability           int64
chlorine           float64
hardness           float64
solids             float64
dtype: object
```

Displaying the number of duplicates:

```
df.shape - df.drop_duplicates().shape
```

```
(0, 0)
```

Displaying a table representing the characteristics of the dataframe when duplicates are deleted:

```
df.drop_duplicates().describe()
```

	ph	sulfate	conductivity	organic_carbon	trihalomethanes	turbidity	potability	chlorine	hardness	solids
count	2785.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000	2822.000000	3276.000000	3276.000000
mean	7.080795	333.775777	426.205111	14.284970	66.396293	3.949111	0.390110	7.234663	196.369496	22014.092526
std	1.594320	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849	1.594320	32.879761	8768.570828
min	0.000000	129.000000	181.483754	2.200000	8.577013	1.450000	0.000000	0.352000	47.432000	320.942611
25%	6.093092	307.699498	365.772042	12.065762	55.844536	3.439711	0.000000	6.127421	176.850538	15666.690297
50%	7.036752	333.073546	423.445884	14.218338	66.622485	3.955028	0.000000	7.130299	196.967627	20927.833607
75%	8.062316	359.948574	481.792304	16.557652	77.337473	4.500320	1.000000	8.114887	216.667456	27332.762127
max	14.000000	481.030642	753.342620	28.300763	124.000000	6.739191	1.000000	10.779375	323.124160	61227.196008

Identify missing data:

```
df.isnull().sum()
```

```
ph                 491
sulfate            781
conductivity         0
organic_carbon       0
trihalomethanes     162
turbidity            0
potability           0
chlorine           158
hardness             0
solids               0
dtype: int64
```

Identify duplicate data:

```
df.duplicated()
```

```
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

Displaying missing values with the mean value of the associated data:

```
df.fillna(df.mean(), inplace=True)
df.head(5)
```

	ph	sulfate	conductivity	organic_carbon	trihalomethanes	turbidity	potability	chlorine	hardness	solids
0	7.080795	268.646941	564.308654	10.379783	86.990970	2.963135	0	7.300212	204.890455	20791.318981
1	3.716080	333.775777	592.885359	15.180013	56.329076	4.500656	0	7.234663	129.422921	18630.057858
2	8.099124	333.775777	418.606213	16.868637	66.420093	3.055934	0	6.635246	224.236259	19909.541732
3	8.316766	304.548754	475.492474	13.688024	100.341674	4.628771	0	9.077200	214.373394	22018.417441
4	9.092223	430.486851	389.375566	12.707760	31.997993	4.075075	0	1.571731	181.101509	17207.402449

Displaying the number of duplicates:

```
df.shape - df.drop_duplicates().shape
```

```
(0, 0)
```

Displaying a table representing the characteristics of the dataframe when duplicates are deleted:

```
df.drop_duplicates().describe()
```

	ph	sulfate	conductivity	organic_carbon	trihalomethanes	turbidity	potability	chlorine	hardness	solids
count	2785.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000	2822.000000	3276.000000	3276.000000
mean	7.080795	333.775777	426.205111	14.284970	66.396293	3.949111	0.390110	7.234663	196.369496	22014.092526
std	1.594320	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849	1.594320	32.879761	8768.570828
min	0.000000	129.000000	181.483754	2.200000	8.577013	1.450000	0.000000	0.352000	47.432000	320.942611
25%	6.093092	307.699498	365.772042	12.065762	55.844536	3.439711	0.000000	6.127421	176.850538	15666.690297
50%	7.036752	333.073546	423.445884	14.218338	66.622485	3.955028	0.000000	7.130299	196.967627	20927.833607
75%	8.062316	359.948574	481.792304	16.557652	77.337473	4.500320	1.000000	8.114887	216.667456	27332.762127
max	14.000000	481.030642	753.342620	28.300763	124.000000	6.739191	1.000000	10.779375	323.124160	61227.196008

Identify duplicate data:

```
df.duplicated()
```

```
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

Data visualization

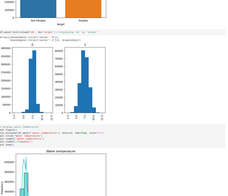
A pair plot for the sulfate and chlorine temperatures on a 2 bar graph:

```
pd.plotting.scatter_matrix(df,figsize=(20,20))
```

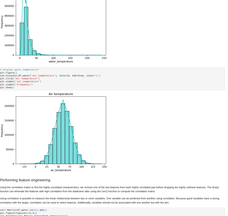
```
array([[<Axis: ph>, <Axis: sulfate>, <Axis: conductivity>,
        <Axis: organic_carbon>, <Axis: trihalomethanes>,
        <Axis: turbidity>, <Axis: potability>, <Axis: chlorine>,
        <Axis: hardness>, <Axis: solids>],
       [<Axis: sulfate>, <Axis: ph>, <Axis: conductivity>,
        <Axis: organic_carbon>, <Axis: trihalomethanes>,
        <Axis: turbidity>, <Axis: potability>, <Axis: chlorine>,
        <Axis: hardness>, <Axis: solids>],
       [<Axis: conductivity>, <Axis: sulfate>, <Axis: ph>,
        <Axis: organic_carbon>, <Axis: trihalomethanes>,
        <Axis: turbidity>, <Axis: potability>, <Axis: chlorine>,
        <Axis: hardness>, <Axis: solids>],
       [<Axis: organic_carbon>, <Axis: sulfate>, <Axis: conductivity>,
        <Axis: ph>, <Axis: trihalomethanes>, <Axis: turbidity>,
        <Axis: potability>, <Axis: chlorine>, <Axis: hardness>,
        <Axis: solids>],
       [<Axis: trihalomethanes>, <Axis: sulfate>, <Axis: conductivity>,
        <Axis: organic_carbon>, <Axis: ph>, <Axis: turbidity>,
        <Axis: potability>, <Axis: chlorine>, <Axis: hardness>,
        <Axis: solids>],
       [<Axis: turbidity>, <Axis: sulfate>, <Axis: conductivity>,
        <Axis: organic_carbon>, <Axis: trihalomethanes>, <Axis: ph>,
        <Axis: potability>, <Axis: chlorine>, <Axis: hardness>,
        <Axis: solids>],
       [<Axis: potability>, <Axis: sulfate>, <Axis: conductivity>,
        <Axis: organic_carbon>, <Axis: trihalomethanes>,
        <Axis: turbidity>, <Axis: ph>, <Axis: chlorine>,
        <Axis: hardness>, <Axis: solids>],
       [<Axis: chlorine>, <Axis: sulfate>, <Axis: conductivity>,
        <Axis: organic_carbon>, <Axis: trihalomethanes>,
        <Axis: turbidity>, <Axis: potability>, <Axis: ph>,
        <Axis: hardness>, <Axis: solids>],
       [<Axis: hardness>, <Axis: sulfate>, <Axis: conductivity>,
        <Axis: organic_carbon>, <Axis: trihalomethanes>,
        <Axis: turbidity>, <Axis: potability>, <Axis: chlorine>,
        <Axis: ph>, <Axis: solids>],
       [<Axis: solids>, <Axis: sulfate>, <Axis: conductivity>,
        <Axis: organic_carbon>, <Axis: trihalomethanes>,
        <Axis: turbidity>, <Axis: potability>, <Axis: chlorine>,
        <Axis: hardness>, <Axis: ph>]]
```

```
df['potability'].value_counts(normalize=True)
```

```
potability
0    0.609891
1    0.390110
Name: proportion, dtype: float64
```

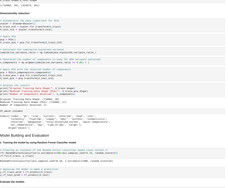



```
df.drop(['potability'],axis=1).hist(bins=50,figsize=(20,15))
```



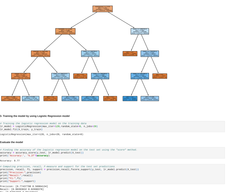

```
df['potability'].value_counts(normalize=True).plot(kind='bar',figsize=(8,5))
```

```
df.drop(['potability'],axis=1).plot(kind='box',subplots=True,layout=(2,5),figsize=(14,6))
```




```
sns.countplot(x='potability',hue='potability',data=df)
```

```
df.groupby('potability').mean().plot(kind='bar',figsize=(8,5))
```




```
df['potability'].value_counts(normalize=True).plot(kind='bar',figsize=(8,5))
```


potability
potability
potability