2.Task - Classification

January 19, 2024

1 2. Task - Classification

- Link to dataset - https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package/
- Since our dataset has labels as classes, either it **will** or **won't** rain, we are gonna use Classification algorithms.
- We are gonna use the dataset used in our 1. Task - Exploration data analysis and clustering (Check it before proceeding in the 2. Task to better understand the dataset).
- Our goal of the classification in this dataset will be to predict, given the features of an example, whether or not it will rain tommorow. The output of the classification model will be True (1), if the model predicts that it will rain tommorow, or False (0) in case it won't rain tommorow.

NOTE: Difference between Regression and Classification tasks is that in Regression we want to predict real value, such as the price of a house, price of a cryptocurrency, etc, whereas in Classification we want to predict classes (e.g. to which class the example falls to, recognize handwritten digits)

1.1 Import packages

[ ]:
```
import pandas as pd                                          # Dataframes
import numpy as np                                           # Matrices and linear␣
 ↪algebra
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=UserWarning)

pd.set_option('display.max_columns', None) # Show all columns
```

1.2 Load dataset

- Load the dataset and take a closer look at the examples

[ ]:
```
pathToDataset = "..\WeatherAUS_Data\weatherAUS.csv"
origDataframe = pd.read_csv(pathToDataset)
origDataframe
```

```
[ ]:            Date Location  MinTemp  MaxTemp  Rainfall  Evaporation  \
     0       2008-12-01   Albury     13.4     22.9       0.6          NaN
     1       2008-12-02   Albury      7.4     25.1       0.0          NaN
     2       2008-12-03   Albury     12.9     25.7       0.0          NaN
     3       2008-12-04   Albury      9.2     28.0       0.0          NaN
     4       2008-12-05   Albury     17.5     32.3       1.0          NaN
     …              …      …      …      …      …       …
     145455  2017-06-21    Uluru      2.8     23.4       0.0          NaN
     145456  2017-06-22    Uluru      3.6     25.3       0.0          NaN
     145457  2017-06-23    Uluru      5.4     26.9       0.0          NaN
     145458  2017-06-24    Uluru      7.8     27.0       0.0          NaN
     145459  2017-06-25    Uluru     14.9      NaN       0.0          NaN

             Sunshine WindGustDir  WindGustSpeed WindDir9am WindDir3pm  \
     0            NaN           W           44.0          W        WNW
     1            NaN         WNW           44.0        NNW        WSW
     2            NaN         WSW           46.0          W        WSW
     3            NaN          NE           24.0         SE          E
     4            NaN           W           41.0        ENE         NW
     …           …          …            …        …       …
     145455       NaN           E           31.0         SE        ENE
     145456       NaN         NNW           22.0         SE          N
     145457       NaN           N           37.0         SE        WNW
     145458       NaN          SE           28.0        SSE          N
     145459       NaN         NaN            NaN        ESE        ESE

             WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  \
     0               20.0          24.0         71.0         22.0       1007.7
     1                4.0          22.0         44.0         25.0       1010.6
     2               19.0          26.0         38.0         30.0       1007.6
     3               11.0           9.0         45.0         16.0       1017.6
     4                7.0          20.0         82.0         33.0       1010.8
     …             …           …          …          …          …
     145455          13.0          11.0         51.0         24.0       1024.6
     145456          13.0           9.0         56.0         21.0       1023.5
     145457           9.0           9.0         53.0         24.0       1021.0
     145458          13.0           7.0         51.0         24.0       1019.4
     145459          17.0          17.0         62.0         36.0       1020.2

             Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm RainToday  \
     0            1007.1       8.0       NaN     16.9     21.8        No
     1            1007.8       NaN       NaN     17.2     24.3        No
     2            1008.7       NaN       2.0     21.0     23.2        No
     3            1012.8       NaN       NaN     18.1     26.5        No
     4            1006.0       7.0       8.0     17.8     29.7        No
     …              …      …      …     …     …       …
     145455       1020.3       NaN       NaN     10.1     22.4        No
```

```
145456       1019.1       NaN       NaN      10.9      24.5         No
145457       1016.8       NaN       NaN      12.5      26.1         No
145458       1016.5       3.0       2.0      15.1      26.0         No
145459       1017.9       8.0       8.0      15.0      20.9         No

       RainTomorrow
0                No
1                No
2                No
3                No
4                No
…               …
145455           No
145456           No
145457           No
145458           No
145459          NaN

[145460 rows x 23 columns]
```

[ ]:
```python
origDataframe.describe()
```

[ ]:
```
             MinTemp        MaxTemp       Rainfall   Evaporation  \
count  143975.000000  144199.000000  142199.000000  82670.000000
mean       12.194034      23.221348       2.360918      5.468232
std         6.398495       7.119049       8.478060      4.193704
min        -8.500000      -4.800000       0.000000      0.000000
25%         7.600000      17.900000       0.000000      2.600000
50%        12.000000      22.600000       0.000000      4.800000
75%        16.900000      28.200000       0.800000      7.400000
max        33.900000      48.100000     371.000000    145.000000

           Sunshine  WindGustSpeed   WindSpeed9am   WindSpeed3pm  \
count  75625.000000  135197.000000  143693.000000  142398.000000
mean       7.611178      40.035230      14.043426      18.662657
std        3.785483      13.607062       8.915375       8.809800
min        0.000000       6.000000       0.000000       0.000000
25%        4.800000      31.000000       7.000000      13.000000
50%        8.400000      39.000000      13.000000      19.000000
75%       10.600000      48.000000      19.000000      24.000000
max       14.500000     135.000000     130.000000      87.000000

             Humidity9am    Humidity3pm   Pressure9am    Pressure3pm  \
count  142806.000000  140953.000000  130395.00000  130432.000000
mean       68.880831      51.539116    1017.64994    1015.255889
std        19.029164      20.795902       7.10653       7.037414
min         0.000000       0.000000     980.50000     977.100000
```

```
25%         57.000000      37.000000     1012.90000     1010.400000
50%         70.000000      52.000000     1017.60000     1015.200000
75%         83.000000      66.000000     1022.40000     1020.000000
max        100.000000     100.000000     1041.00000     1039.600000

             Cloud9am       Cloud3pm         Temp9am        Temp3pm
count    89572.000000   86102.000000   143693.000000   141851.00000
mean         4.447461       4.509930       16.990631       21.68339
std          2.887159       2.720357        6.488753        6.93665
min          0.000000       0.000000       -7.200000       -5.40000
25%          1.000000       2.000000       12.300000       16.60000
50%          5.000000       5.000000       16.700000       21.10000
75%          7.000000       7.000000       21.600000       26.40000
max          9.000000       9.000000       40.200000       46.70000
```

1.3 Import few different Classifiers

We are gonna use few different classifiers, such as:

- Logistic Regression (Similar to Linear regression, but its output is in range 0 to 1 (after applying threshold, the outpur is either class 1 or 0), so pretty much Binary classification)
- SGD Classifier (Logistic regression implemented with Stochastic Gradient Descent)
- Decision Tree Classifier (Decision Tree Classifier)
- Random Forest Classifier (Uses N Decision Tree Classifiers to compute the output of the Random Forest Classifier. The majority class is predicted)
- MLP Classifier (Multi-Layer Perceptron which is considered as a simple Neural Network)

[ ]:
```
from sklearn.linear_model import LogisticRegression # Load Logistic regression␣
↪model (used for classification)
from sklearn.linear_model import SGDClassifier      # Classification algorithm␣
↪using SGD algorithm for training
from sklearn.tree import DecisionTreeClassifier     # Decision tree classifier
from sklearn.ensemble import RandomForestClassifier # Rnadom forest classifier␣
↪(uses multiple Decision Tree Classifiers to compute the output)
from sklearn.neural_network import MLPClassifier    # Multi-Layer classifier
```

1.3.1 Handle categorical columns and NaN values

Before we can start training our models, we need to prepare the training set, meaning to preprocess the training set (Change Categorical features to numerical, Handle NaN values in features) and separate the labels from the train set

Convert categorical columns

- One of the way to convert categorical columns to numerical, we can use factorize on every categorical column in the Dataframe

[ ]:
```
copyDataframe = origDataframe.copy()

categoricalColumns = origDataframe.select_dtypes(['object']).columns
copyDataframe[categoricalColumns] = copyDataframe[categoricalColumns].
↪apply(lambda x: pd.factorize(x)[0])

copyDataframe
```

[ ]:
```
        Date  Location  MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  \
0          0         0     13.4     22.9       0.6          NaN       NaN
1          1         0      7.4     25.1       0.0          NaN       NaN
2          2         0     12.9     25.7       0.0          NaN       NaN
3          3         0      9.2     28.0       0.0          NaN       NaN
4          4         0     17.5     32.3       1.0          NaN       NaN
…        …         …        …        …        …          …         …
145455  3035        48      2.8     23.4       0.0          NaN       NaN
145456  3036        48      3.6     25.3       0.0          NaN       NaN
145457  3037        48      5.4     26.9       0.0          NaN       NaN
145458  3038        48      7.8     27.0       0.0          NaN       NaN
145459  3039        48     14.9      NaN       0.0          NaN       NaN

        WindGustDir  WindGustSpeed  WindDir9am  WindDir3pm  WindSpeed9am  \
0                 0           44.0           0           0          20.0
1                 1           44.0           1           1           4.0
2                 2           46.0           0           1          19.0
3                 3           24.0           2           2          11.0
4                 0           41.0           3           3           7.0
…               …              …           …           …             …
145455           14           31.0           2           7          13.0
145456            4           22.0           2          12          13.0
145457            5           37.0           2           0           9.0
145458           12           28.0           5          12          13.0
145459           -1            NaN          11           6          17.0

        WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  Pressure3pm  \
0               24.0         71.0         22.0       1007.7       1007.1
1               22.0         44.0         25.0       1010.6       1007.8
2               26.0         38.0         30.0       1007.6       1008.7
3                9.0         45.0         16.0       1017.6       1012.8
4               20.0         82.0         33.0       1010.8       1006.0
…                …            …            …            …            …
145455          11.0         51.0         24.0       1024.6       1020.3
145456           9.0         56.0         21.0       1023.5       1019.1
145457           9.0         53.0         24.0       1021.0       1016.8
145458           7.0         51.0         24.0       1019.4       1016.5
145459          17.0         62.0         36.0       1020.2       1017.9
```

```
        Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday  RainTomorrow
0            8.0       NaN     16.9     21.8          0             0
1            NaN       NaN     17.2     24.3          0             0
2            NaN       2.0     21.0     23.2          0             0
3            NaN       NaN     18.1     26.5          0             0
4            7.0       8.0     17.8     29.7          0             0
…          …       …      …      …      …        …
145455       NaN       NaN     10.1     22.4          0             0
145456       NaN       NaN     10.9     24.5          0             0
145457       NaN       NaN     12.5     26.1          0             0
145458       3.0       2.0     15.1     26.0          0             0
145459       8.0       8.0     15.0     20.9          0            -1

[145460 rows x 23 columns]
```

Clean set from NaN values

- To handle NaN and empty column values, I am going to drop those rows from the training set. It will have huge impact on the training and generally this options is not the best when preprocessing data.

[ ]:
```
print(f"Shape before dropwing rows: {copyDataframe.shape}")

copyDataframe = copyDataframe.dropna().copy()
print(f"Shape after dropwing rows: {copyDataframe.shape}")
```

```
Shape before dropwing rows: (145460, 23)
Shape after dropwing rows: (58236, 23)
```

NOTE: As we can see, due to this drastic method, we dropped nearly 100 000 rows from the training set, that's about 2/3. Also we lost a lot of information.

2 Prepare training set for training

- Separate features and labels

[ ]:
```
labels = pd.DataFrame(copyDataframe['RainTomorrow'])
features = copyDataframe.drop('RainTomorrow', axis=1)
print(labels)
print(features)
```

```
        RainTomorrow
6049               0
6050               0
6052               0
6053               0
6054               0
…                …
142298             0
```

```
142299            0
142300            0
142301            0
142302            0

[58236 rows x 1 columns]
        Date  Location  MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  \
6049      31         2     17.9     35.2       0.0         12.0      12.3
6050      32         2     18.4     28.9       0.0         14.8      13.0
6052      34         2     19.4     37.6       0.0         10.8      10.6
6053      35         2     21.9     38.4       0.0         11.4      12.2
6054      36         2     24.2     41.0       0.0         11.2       8.4
…       …       …       …        …        …         …        …
142298  3034        46     19.3     33.4       0.0          6.0      11.0
142299  3035        46     21.2     32.6       0.0          7.6       8.6
142300  3036        46     20.7     32.8       0.0          5.6      11.0
142301  3037        46     19.5     31.8       0.0          6.2      10.6
142302  3038        46     20.2     31.7       0.0          5.6      10.7

        WindGustDir  WindGustSpeed  WindDir9am  WindDir3pm  WindSpeed9am  \
6049             15           48.0           3          10           6.0
6050             10           37.0           5           5          19.0
6052              6           46.0          15           8          30.0
6053              1           31.0          14           1           6.0
6054              1           35.0          13           0          17.0
…              …             …          …          …          …
142298            8           35.0           2          15           9.0
142299           14           37.0           2          11          13.0
142300           14           33.0          12           4          17.0
142301           13           26.0           2           8           9.0
142302            8           30.0           3           8          15.0

        WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  Pressure3pm  \
6049            20.0         20.0         13.0       1006.3       1004.4
6050            19.0         30.0          8.0       1012.9       1012.1
6052            15.0         42.0         22.0       1012.3       1009.2
6053             6.0         37.0         22.0       1012.7       1009.1
6054            13.0         19.0         15.0       1010.7       1007.4
…              …          …          …          …          …
142298          20.0         63.0         32.0       1013.9       1010.5
142299          11.0         56.0         28.0       1014.6       1011.2
142300          11.0         46.0         23.0       1015.3       1011.8
142301          17.0         62.0         58.0       1014.9       1010.7
142302           7.0         73.0         32.0       1013.9       1009.7

        Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday
6049         2.0       5.0     26.6     33.4          0
6050         1.0       1.0     20.3     27.0          0
```

```
6052          1.0      6.0     28.7     34.9          0
6053          1.0      5.0     29.1     35.6          0
6054          1.0      6.0     33.6     37.6          0
…           …       …       …       …       …
142298        0.0      1.0     24.5     32.3          0
142299        7.0      0.0     24.8     32.0          0
142300        0.0      0.0     24.8     32.1          0
142301        1.0      1.0     24.8     29.2          0
142302        6.0      5.0     25.4     31.0          0

[58236 rows x 22 columns]
```

- Scale data using Standardization metods or normalization methods

[ ]:
```
from sklearn.preprocessing import StandardScaler

standardScaler = StandardScaler()
standardScaler.fit_transform(features)
```

[ ]:
```
array([[-1.58286518, -1.87728304,  0.70599206, …,  1.2903242 ,
         1.57666339, -0.53132627],
       [-1.58171141, -1.87728304,  0.78332492, …,  0.33632541,
         0.64036635, -0.53132627],
       [-1.57940387, -1.87728304,  0.93799064, …,  1.60832379,
         1.79610801, -0.53132627],
       …,
       [ 1.88421016,  1.60023354,  1.13905608, …,  1.01775311,
         1.38647805, -0.53132627],
       [ 1.88536393,  1.60023354,  0.95345721, …,  1.01775311,
         0.96221846, -0.53132627],
       [ 1.8865177 ,  1.60023354,  1.06172322, …,  1.10861014,
         1.225552  , -0.53132627]])
```

- Create train and test set (Also for further learning, we can use cross-validation set)

NOTE: Row count must be the same after the split action!

[ ]:
```
from sklearn.model_selection import train_test_split

trainFeatures, testFeatures = train_test_split(features, test_size=0.3)
print(f"Train features shape: {trainFeatures.shape}")
print(f"Test features shape: {testFeatures.shape}\n")

# Do the same with the labels
trainLabels, testLabels = train_test_split(labels, test_size=0.3)
print(f"Train features shape: {trainLabels.shape}")
print(f"Test features shape: {testLabels.shape}")
```

```
Train features shape: (40765, 22)
Test features shape: (17471, 22)
```

```
Train features shape: (40765, 1)
Test features shape: (17471, 1)
```

3 Train the Classifiers

- We will train the models (Logistic Regression, SGD Classifier, Decision Tree Classifier, Random Forest Classifier, MLP Classifier) using fit method.
- Compute the score of each model and compare them.
- After that, we are gonna evaluate them on the test set to see how well/badly they generalize.
- We want to develop a Machine Learning algorithm which will perform well on the new unseed data, not just on the training set.

[ ]:
```python
models = [
    ["LogisticRegression", LogisticRegression()],
    ["SGDClassifier", SGDClassifier()],
    ["DecisionTreeClassifier", DecisionTreeClassifier()],
    ["RandomForestClassifier", RandomForestClassifier()],
    ["MLPClassifier", MLPClassifier()]
]

print("TRAIN set scores after training:")
for name, model in models:
    # Train the model
    model.fit(trainFeatures, trainLabels.values.ravel())

    # Compute the score for that model and store it
    score = model.score(trainFeatures, trainLabels)

    # Print the score
    print(f"Score of a {name} is: {score}")
```

```
TRAIN set scores after training:
Score of a LogisticRegression is: 0.7797129890837728
Score of a SGDClassifier is: 0.21778486446706732
Score of a DecisionTreeClassifier is: 1.0
Score of a RandomForestClassifier is: 1.0
Score of a MLPClassifier is: 0.557806942229854
```

4 Evaluate the classifiers

- To evaluate the model, we need to use test features and test labels.

NOTE: Don't forget to standardize the test set using the same standardization method used on the training set (In my case, they are already standardized, because I used the standardization method before splitting the data to sets).

[ ]:
```
print("TEST set scores")
for name, model in models:
    # Compute the score for that model and store it
    score = model.score(trainFeatures, trainLabels)

    print(f"Score of a {name} is: {score}")
```

```
TEST set scores
Score of a LogisticRegression is: 0.7797129890837728
Score of a SGDClassifier is: 0.21778486446706732
Score of a DecisionTreeClassifier is: 1.0
Score of a RandomForestClassifier is: 1.0
Score of a MLPClassifier is: 0.557806942229854
```

5 Predict using the trained classifiers

- Let's predict some outcomes (for simplicity I'll use only Logistic Regressor to do the job)
  - Example from the Training set
  - Example from the Test set
- Compare the predicted values with the truth labels

[ ]:
```
randomIndex = 4354
logisticRegression = models[0][1]

# Prepare examples
exampleFromTrainSet = np.array(trainFeatures.iloc[randomIndex]).reshape(1, -1)
exampleFromTestSet =  np.array(testFeatures.iloc[randomIndex]) .reshape(1, -1)

# Prepare its truth values
exampleLabelFromTrainSet = np.array(trainLabels.iloc[randomIndex])
exampleLabelFromTestSet = np.array(testLabels.iloc[randomIndex])

# Prediction of an example from Train set
predicted = logisticRegression.predict(exampleFromTrainSet)

print(f"TRAIN SET, example: {exampleFromTrainSet}")
print(f"Predicted label: {predicted}")
print(f"Truth value: {exampleLabelFromTrainSet}\n")

# Prediction of an example from Test set
predicted = logisticRegression.predict(exampleFromTestSet)
print(f"TRAIN SET, example: {exampleFromTestSet}")
print(f"Predicted label: {predicted}")
print(f"Truth value: {exampleLabelFromTestSet}")
```

```
TRAIN SET, example: [[1.1260e+03 4.6000e+01 2.4000e+01 2.8000e+01 4.4600e+01
```

```
3.2000e+00
  0.0000e+00 7.0000e+00 6.3000e+01 1.4000e+01 4.0000e+00 3.0000e+01
  2.0000e+01 8.8000e+01 9.5000e+01 1.0071e+03 1.0059e+03 7.0000e+00
  8.0000e+00 2.6900e+01 2.4700e+01 1.0000e+00]]
Predicted label: [0]
Truth value: [0]

TRAIN SET, example: [[2.2770e+03 2.8000e+01 1.2700e+01 2.3500e+01 0.0000e+00
4.6000e+00
  9.3000e+00 9.0000e+00 2.0000e+01 4.0000e+00 6.0000e+00 4.0000e+00
  1.1000e+01 6.9000e+01 4.9000e+01 1.0309e+03 1.0276e+03 1.0000e+00
  7.0000e+00 1.8300e+01 2.2200e+01 0.0000e+00]]
Predicted label: [0]
Truth value: [0]
```

6 Summary

In this notebook, we ran classification algorithm on the given dataset (from 1. Task), Specifically we:

- Load data from dataset
- Preprocessed data (Cleaning them, standardizing them, splitted them to train/test sets)
- Run few different classification algorithms (Logistic regression, SGD Classifier, Decision Tree Classifier, Random Forest Classifier, MLP Classifier)
- Evaluate them and used them to make predictions