Final Project: Classification with Python

Table of Contents

Instructions

In this notebook, you will practice all the classification algorithms that we have learned in this course.

Below, is where we are going to use the classification algorithms to create a model based on our training data and evaluate our testing data using evaluation metrics learned in the course.

We will use some of the algorithms taught in the course, specifically:

1. Linear Regression
2. KNN
3. Decision Trees
4. Logistic Regression
5. SVM

We will evaluate our models using:

1. Accuracy Score
2. Jaccard Index
3. F1-Score
4. LogLoss

5. Mean Absolute Error
6. Mean Squared Error
7. R2-Score

About The Dataset

The original source of the data is Australian Government's Bureau of Meteorology and the latest data can be gathered from http://www.bom.gov.au/climate/dwo/.

The dataset to be used has extra columns like 'RainToday' and our target is 'RainTomorrow', which was gathered from the Rattle at https://bitbucket.org/kayontoga/rattle/src/master/data/weatherAUS.RData

This dataset contains observations of weather metrics for each day from 2008 to 2017. The **weatherAUS.csv** dataset includes the following fields:

Field	Description	Unit	Type
Date	Date of the Observation in YYYY-MM-DD	Date	object
Location	Location of the Observation	Location	object
MinTemp	Minimum temperature	Celsius	float
MaxTemp	Maximum temperature	Celsius	float
Rainfall	Amount of rainfall	Millimeters	float
Evaporation	Amount of evaporation	Millimeters	float
Sunshine	Amount of bright sunshine	hours	float
WindGustDir	Direction of the strongest gust	Compass Points	object
WindGustSpeed	Speed of the strongest gust	Kilometers/Hour	object
WindDir9am	Wind direction averaged of 10 minutes prior to 9am	Compass Points	object
WindDir3pm	Wind direction averaged of 10 minutes prior to 3pm	Compass Points	object
WindSpeed9am	Wind speed averaged of 10 minutes prior to 9am	Kilometers/Hour	float
WindSpeed3pm	Wind speed averaged of 10 minutes prior to 3pm	Kilometers/Hour	float
Humidity9am	Humidity at 9am	Percent	float
Humidity3pm	Humidity at 3pm	Percent	float
Pressure9am	Atmospheric pressure reduced to mean sea level at 9am	Hectopascal	float
Pressure3pm	Atmospheric pressure reduced to mean sea level at 3pm	Hectopascal	float
Cloud9am	Fraction of the sky obscured by cloud at 9am	Eights	float
Cloud3pm	Fraction of the sky obscured by cloud at 3pm	Eights	float
Temp9am	Temperature at 9am	Celsius	float
Temp3pm	Temperature at 3pm	Celsius	float
RainToday	If there was rain today	Yes/No	object
RISK_MM	Amount of rain tomorrow	Millimeters	float

Field	Description	Unit	Type
RainTomorrow	If there is rain tomorrow	Yes/No	float

Column definitions were gathered from http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml

Import the required libraries

In [ ]:
```
# All Libraries required for this lab are listed below. The libraries pre-install
!pip install -qy pandas==1.3.4 numpy==1.21.4 seaborn==0.9.0 matplotlib==3.5.0 sci
```

In [2]:
```
# Surpress warnings:
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn
```

In [3]:
```
#you are running the lab in your  browser, so we will install the libraries using
import piplite
await piplite.install(['pandas'])
await piplite.install(['numpy'])
```

In [4]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.metrics import jaccard_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
from sklearn.metrics import confusion_matrix, accuracy_score
import sklearn.metrics as metrics
```

Importing the Dataset

In [5]:
```
from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())
```

In [6]:
```
path='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDevel
```

In [7]:
```
await download(path, "Weather_Data.csv")
filename ="Weather_Data.csv"
```

In [13]:
```
df = pd.read_csv("Weather_Data.csv")
df.head()
```

Out[13]:

	Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpee
0	2/1/2008	19.5	22.4	15.6	6.2	0.0	W	4
1	2/2/2008	19.5	25.6	6.0	3.4	2.7	W	4
2	2/3/2008	21.6	24.5	6.6	2.4	0.1	W	4
3	2/4/2008	20.2	22.8	18.8	2.2	0.0	W	4
4	2/5/2008	19.7	25.7	77.4	4.8	0.0	W	4

5 rows × 22 columns

Data Preprocessing

One Hot Encoding

First, we need to perform one hot encoding to convert categorical variables to binary variables.

In [14]:
```
df_sydney_processed = pd.get_dummies(data=df, columns=['RainToday', 'WindGustDir'
```

In [15]:
```
df_sydney_processed.Date
```

Out[15]:
```
0        2/1/2008
1        2/2/2008
2        2/3/2008
3        2/4/2008
4        2/5/2008
           ...
3266    6/21/2017
3267    6/22/2017
3268    6/23/2017
3269    6/24/2017
3270    6/25/2017
Name: Date, Length: 3271, dtype: object
```

Next, we replace the values of the 'RainTomorrow' column changing them from a categorical column to a binary column. We do not use the `get_dummies` method because we would end up with two columns for 'RainTomorrow' and we do not want, since 'RainTomorrow' is our target.

In [16]:
```
df_sydney_processed.replace(['No', 'Yes'], [0,1], inplace=True)
```

In [17]:
```
df_sydney_processed.head(2)
```

Out[17]:

	Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed
0	2/1/2008	19.5	22.4	15.6	6.2	0.0	41	
1	2/2/2008	19.5	25.6	6.0	3.4	2.7	41	

2 rows × 68 columns

Training Data and Test Data

Now, we set our 'features' or x values and our Y or target variable.

In [18]:
```
df_sydney_processed.drop('Date',axis=1,inplace=True)
```

In [19]:
```
df_sydney_processed = df_sydney_processed.astype(float)
```

In [20]:
```
features = df_sydney_processed.drop(columns='RainTomorrow', axis=1)
Y = df_sydney_processed['RainTomorrow']
```

Linear Regression

Use the `train_test_split` function to split the `features` and `Y` dataframes with a `test_size` of `0.2` and the `random_state` set to `10`.

In [21]:
```
x_train, x_test, y_train, y_test = train_test_split(features, Y, test_size=0.2, r
```

Create and train a Linear Regression model called LinearReg using the training data (`x_train`, `y_train`).

In [22]:
```
LinearReg = LinearRegression()
LinearReg.fit(x_train, y_train)
```

Out[22]:
```
LinearRegression()
```

Use the `predict` method on the testing data (`x_test`) and save it to the array `predictions`.

In [23]:
```
predictions = LinearReg.predict(x_test)
```

Using the `predictions` and the `y_test` dataframe calculate the value for each metric using the appropriate function.

In [24]:
```
from sklearn.metrics import r2_score
LinearRegression_MAE = np.mean(np.absolute(y_test - predictions))
LinearRegression_MSE = np.mean((y_test - predictions)**2)
LinearRegression_R2 = r2_score(y_test, predictions)
```

Show the MAE, MSE, and R2 in a tabular format using data frame for the linear model.

In [25]:
```
Report = pd.DataFrame({'MAE': [LinearRegression_MAE],'MSE': [LinearRegression_MSE
```

In [26]:
```
Report
```

Out[26]:

	MAE	MSE	R_SQUARE
0	0.26813	0.122921	0.38946

KNN

Create and train a KNN model called KNN using the training data (`x_train`, `y_train`) with the `n_neighbors` parameter set to `4`.

In [27]:
```
KNN = KNeighborsClassifier(n_neighbors = 4).fit(x_train,y_train)
```

Now use the `predict` method on the testing data (`x_test`) and save it to the array `predictions`.

In [28]:
```
predictions = KNN.predict(x_test)
```

Using the `predictions` and the `y_test` dataframe calculate the value for each metric using the appropriate function.

In [29]:
```
KNN_Accuracy_Score = metrics.accuracy_score(y_test, predictions)
KNN_JaccardIndex = metrics.jaccard_score(y_test, predictions)
KNN_F1_Score = metrics.f1_score(y_test, predictions)
```

Decision Tree

Create and train a Decision Tree model called Tree using the training data (`x_train`, `y_train`).

In [30]:
```
Tree = DecisionTreeClassifier().fit(x_train, y_train)
```

Now use the `predict` method on the testing data (`x_test`) and save it to the array `predictions`.

In [31]:
```
predictions = Tree.predict(x_test)
```

Using the `predictions` and the `y_test` dataframe calculate the value for each metric using the appropriate function.

In [32]:
```
Tree_Accuracy_Score = metrics.accuracy_score(predictions, y_test)
Tree_JaccardIndex = metrics.jaccard_score(predictions, y_test)
Tree_F1_Score = metrics.f1_score(predictions, y_test)
```

Logistic Regression

Use the `train_test_split` function to split the `features` and `Y` dataframes with a `test_size` of `0.2` and the `random_state` set to `1`.

In [33]:
```
x_train, x_test, y_train, y_test = train_test_split(features, Y, test_size=0.2, r
```

Create and train a LogisticRegression model called LR using the training data (`x_train`, `y_train`) with the `solver` parameter set to `liblinear`.

In [34]:
```
LR = LogisticRegression().fit(x_train, y_train)
```

Now, use the `predict` method on the testing data (`x_test`) and save it to the array `predictions`.

In [35]:
```
predictions = LR.predict(x_test)
```

Using the `predictions` and the `y_test` dataframe calculate the value for each metric using the appropriate function.

In [36]:
```
LR_Accuracy_Score = metrics.accuracy_score(predictions, y_test)
LR_JaccardIndex = metrics.jaccard_score(predictions, y_test)
LR_F1_Score = metrics.f1_score(predictions, y_test)
LR_Log_Loss = metrics.log_loss(predictions, y_test)
```

SVM

Create and train a SVM model called SVM using the training data (`x_train`, `y_train`).

In [37]:
```
from sklearn import svm
SVM = SVM = svm.SVC(kernel='rbf').fit(x_train, y_train)
```

Now use the `predict` method on the testing data (`x_test`) and save it to the array `predictions`.

In [38]:
```
predictions = SVM.predict(x_test)
```

Using the `predictions` and the `y_test` dataframe calculate the value for each metric using the appropriate function.

In [39]:
```
SVM_Accuracy_Score = metrics.accuracy_score(predictions, y_test)
SVM_JaccardIndex = metrics.jaccard_score(predictions, y_test)
SVM_F1_Score = metrics.f1_score(predictions, y_test)
```

Report

Show the Accuracy,Jaccard Index,F1-Score and LogLoss in a tabular format using data frame for all of the above models.

*LogLoss is only for Logistic Regression Model

In [40]:
```
Report = pd.DataFrame({'Accuracy_score': [SVM_Accuracy_Score],'JaccardIndex': [SV
```

In [41]:
```
Report
```

Out[41]:

	Accuracy_score	JaccardIndex	F1_Score
0	0.722137	0.0	0.0