Skills Network
(https://skills.network/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-2022-01-01)

Final Project: Classification with Python

Table of Contents

Estimated Time Needed: **180 min**

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

Finally, you will use your models to generate the report at the end.

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/ (http://www.bom.gov.au/climate/dwo/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NSkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-2022-01-01).

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 (https://bitbucket.org/kayontoga/rattle/src/master/data/weatherAUS.RData?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NSkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-2022-01-01)

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

Field	Description	Unit	Type
RainToday	If there was rain today	Yes/No	object
RISK_MM	Amount of rain tomorrow	Millimeters	float
RainTomorrow	If there is rain tomorrow	Yes/No	float

Column definitions were gathered from http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml (http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NSkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-

Import the required libraries

In [98]:

```
# All Libraries required for this lab are listed below. The libraries pre-installed
# !mamba install -qy pandas==1.3.4 numpy==1.21.4 seaborn==0.9.0 matplotlib==3.5.0 sc
# Note: If your environment doesn't support "!mamba install", use "!pip install"
```

In [99]:

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

In [100]:

```
#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 [101]:

```
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 [102]:

```
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 [103]:

```
path='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDevelope
```

In [104]:

```
await download(path, "Weather_Data.csv")
filename ="Weather_Data.csv"
```

In [105]:

```
df = pd.read_csv("Weather_Data.csv")
df.head()
```

Out[105]:

	Date	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed
0	2/1/2008	19.5	22.4	15.6	6.2	0.0	W	41
1	2/2/2008	19.5	25.6	6.0	3.4	2.7	W	41
2	2/3/2008	21.6	24.5	6.6	2.4	0.1	W	41
3	2/4/2008	20.2	22.8	18.8	2.2	0.0	W	41
4	2/5/2008	19.7	25.7	77.4	4.8	0.0	W	41

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 [106]:

```
df_sydney_processed = pd.get_dummies(data=df, columns=['RainToday', 'WindGustDir', '
```

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 [107]:

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

Training Data and Test Data

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

In [108]:

```
df_sydney_processed.drop('Date',axis=1,inplace=True)
```

In [109]:

```
df_sydney_processed = df_sydney_processed.astype(float)
```

In [110]:

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

Linear Regression

Q1) 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 [111]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [112]:

```
x_train, x_test, y_train, y_test = train_test_split( features,Y, test_size=0.2, ranc
```

Q2) Create and train a Linear Regression model called LinearReg using the training data (`x_train`, `y_train`).

In [113]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [114]:

```
LinearReg = LinearRegression()
LinearReg.fit (x_train, y_train)
print ('Coefficients are: ', LinearReg.coef_)
```

```
Coefficients are:  [-2.36862502e-02  1.30060400e-02  7.29929096e-04
6.49363254e-03
 -3.51643494e-02  4.23733388e-03  1.82788340e-03  7.90075624e-04
  9.56782146e-04  8.55986210e-03  7.69992241e-03 -9.24589847e-03
 -8.88017645e-03  1.00487910e-02  1.44675206e-02 -3.48703168e-03
  8.47590247e+08  8.47590247e+08 -6.41324526e+09 -6.41324526e+09
 -6.41324526e+09 -6.41324526e+09 -6.41324526e+09 -6.41324526e+09
 -6.41324526e+09 -6.41324526e+09 -6.41324526e+09 -6.41324526e+09
 -6.41324526e+09 -6.41324526e+09 -6.41324526e+09 -6.41324526e+09
 -6.41324526e+09 -6.41324526e+09  1.43257002e+10  1.43257002e+10
  1.43257002e+10  1.43257002e+10  1.43257002e+10  1.43257002e+10
  1.43257002e+10  1.43257002e+10  1.43257002e+10  1.43257002e+10
  1.43257002e+10  1.43257002e+10  1.43257002e+10  1.43257002e+10
  1.43257002e+10  1.43257002e+10 -1.09414185e+10 -1.09414185e+10
 -1.09414185e+10 -1.09414185e+10 -1.09414185e+10 -1.09414185e+10
 -1.09414185e+10 -1.09414185e+10 -1.09414185e+10 -1.09414185e+10
 -1.09414185e+10 -1.09414185e+10 -1.09414185e+10 -1.09414185e+10
 -1.09414185e+10 -1.09414185e+10]
```

Q3) Now use the `predict` method on the testing data (`x_test`) and save it to the array `predictions` .

In [115]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [116]:

```
predictions = LinearReg.predict(x_test)
print("Predictions are :",  predictions[:5])
```

```
Predictions are : [0.13187027 0.27623177 0.97818375 0.28743553 0.13241
768]
```

Q4) Using the `predictions` and the `y_test` dataframe calculate the value for each metric using the appropriate function.

In [117]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [118]:

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

print("Mean absolute error : %.3f" % LinearRegression_MAE)
print("Residual sum of squares (MSE) : %.3f" % LinearRegression_MSE)
print("R2 score : %.3f" %LinearRegression_R2 )
```

```
Mean absolute error : 0.256
Residual sum of squares (MSE) : 0.116
R2 score : 0.427
```

Q5) Show the MAE, MSE, and R2 in a tabular format using data frame for the linear model.

In [119]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [120]:

```
data = {'MAE':[LinearRegression_MAE],
     'MSE':[LinearRegression_MSE],
     'R2':[LinearRegression_R2]}
Report =  pd.DataFrame(data, index = ['values'])
Report
```

Out[120]:

	MAE	MSE	R2
values	0.256319	0.11572	0.427133

KNN

Q6) 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 [121]:

```
#Enter Your Code Below, Execute, and Save the Screenshot of the Final Output
```

In [122]:

```
from sklearn.neighbors import KNeighborsClassifier

# creating the KNN model
KNN = KNeighborsClassifier(n_neighbors=4)

# training the KNN model
KNN.fit(x_train, y_train)
```

Out[122]:

```
KNeighborsClassifier(n_neighbors=4)
```

Q7) Now use the `predict` method on the testing data (`x_test`) and save it to the array `predictions` .

In [123]:

```
#Enter Your Code Below, Execute, and Save the Screenshot of the Final Output
```

In [124]:

```
# making predictions on the testing data
predictions = KNN.predict(x_test)
print ('predictions', predictions[0:8])
```

```
predictions [0. 0. 1. 0. 0. 0. 0. 1.]
```

Q8) Using the `predictions` and the `y_test` dataframe calculate the value for each metric using the appropriate function.

In [125]:

```
#Enter Your Code Below, Execute, and Save the Screenshot of the Final Output
```

In [126]:

```
from sklearn.metrics import accuracy_score, jaccard_score, f1_score

# calculating accuracy score
KNN_Accuracy_Score = accuracy_score(y_test, predictions)
print('KNN_Accuracy_Score: %.3f' %KNN_Accuracy_Score)

# calculating Jaccard similarity score
KNN_JaccardIndex = jaccard_score(y_test, predictions, average='weighted')
print('KNN_JaccardIndex: %.3f' %KNN_JaccardIndex)

# calculating F1 score
KNN_F1_Score = f1_score(y_test, predictions, average='weighted')
print ('KNN_F1_Score:  %.3f' %KNN_F1_Score)
```

```
KNN_Accuracy_Score: 0.818
KNN_JaccardIndex: 0.688
KNN_F1_Score:  0.802
```

Decision Tree

Q9) Create and train a Decision Tree model called Tree using the training data (`x_train` , `y_train`).

In [127]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [128]:

```
from sklearn.tree import DecisionTreeClassifier

# creating the Decision Tree model
Tree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)

# training the Decision Tree model
Tree.fit(x_train, y_train)
```

Out[128]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

Q10) Now use the `predict` method on the testing data (`x_test`) and save it to the array `predictions` .

In [129]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [130]:

```
# making predictions on the testing data
predictions = Tree.predict(x_test)
print ('Predicitions: ', predictions[0:8])
```

```
Predicitions:  [0. 0. 1. 0. 0. 0. 0. 1.]
```

Q11) Using the `predictions` and the `y_test` dataframe calculate the value for each metric using the appropriate function.

In [131]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [132]:

```
from sklearn.metrics import accuracy_score, jaccard_score, f1_score

# calculating accuracy score
Tree_Accuracy_Score = accuracy_score(y_test, predictions)
print ('Tree_Accuracy_Score: %.3f' %Tree_Accuracy_Score)

# calculating Jaccard similarity score
Tree_JaccardIndex = jaccard_score(y_test, predictions, average='weighted')
print ('Tree_JaccardIndex: %.3f' %Tree_JaccardIndex)

# calculating F1 score
Tree_F1_Score = f1_score(y_test, predictions, average='weighted')
print ('Tree_F1_Score: %.3f' %Tree_F1_Score)
```

```
Tree_Accuracy_Score: 0.818
Tree_JaccardIndex: 0.697
Tree_F1_Score: 0.813
```

Logistic Regression

Q12) 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 [133]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [134]:

```
from sklearn.model_selection import train_test_split

# spliting the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(features, Y, test_size=0.2, ranc
```

Q13) Create and train a LogisticRegression model called LR using the training data (`x_train`, `y_train`) with the `solver` parameter set to `liblinear`.

In [135]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [136]:

```
from sklearn.linear_model import LogisticRegression

# creating a Logistic Regression model
LR = LogisticRegression(solver='liblinear')

# training the model on the training data
LR.fit(x_train, y_train)
```

Out[136]:

```
LogisticRegression(solver='liblinear')
```

Q14) Now, use the `predict` method on the testing data (`x_test`) and save it to the array `predictions`.

In [137]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [138]:

```
predictions = LR.predict(x_test)
print ('Predictions', predictions[:8])
```

```
Predictions [0. 0. 0. 0. 0. 1. 0. 0.]
```

Q15) Using the `predictions` and the `y_test` dataframe calculate the value for each metric using the appropriate function.

In [139]:

```
#Enter Your Code, Execute and take the Screenshot
```

In [140]:

```
from sklearn.metrics import accuracy_score, jaccard_score, f1_score

# calculating accuracy score
LR_Accuracy_Score = accuracy_score(y_test, predictions)

# calculating Jaccard index
LR_JaccardIndex = jaccard_score(y_test, predictions)

# calculating F1 score
LR_F1_Score = f1_score(y_test, predictions)

# calculating log loss
LR_Log_Loss = log_loss(y_test, LR.predict_proba(x_test))


# printing the results with 3 decimal point accuracy
print("Accuracy Score: {:.3f}".format(LR_Accuracy_Score))
print("Jaccard Index: {:.3f}".format(LR_JaccardIndex))
print("F1 Score: {:.3f}".format(LR_F1_Score))
print("Log Loss: {:.3f}".format(LR_Log_Loss))
```

```
Accuracy Score: 0.835
Jaccard Index: 0.505
F1 Score: 0.671
Log Loss: 0.378
```

SVM

Q16) Create and train a SVM model called SVM using the training data (`x_train` , `y_train`).

In [141]:

```
#Enter Your Code Below, Execute, and Save the Screenshot of the Final Output
```

In [142]:

```
from sklearn import svm

# creating a SVM model
SVM = svm.SVC(kernel='linear')

# fitting the model to the training data
SVM.fit(x_train, y_train)
```

Out[142]:

```
SVC(kernel='linear')
```

Q17) Now use the `predict` method on the testing data (`x_test`) and save it to the array `predictions` .

In [143]:

```
#Enter Your Code Below, Execute, and Save the Screenshot of the Final Output
```

In [144]:

```
# using the predict method on the testing data
predictions = SVM.predict(x_test)
```

Q18) Using the `predictions` and the `y_test` dataframe calculate the value for each metric using the appropriate function.

In [145]:

```
from sklearn.metrics import accuracy_score, jaccard_score, f1_score

# calculating accuracy score
SVM_Accuracy_Score = accuracy_score(y_test, predictions)

# calculating Jaccard index
SVM_JaccardIndex = jaccard_score(y_test, predictions)

# calculating F1 score
SVM_F1_Score = f1_score(y_test, predictions, average='weighted')

# print the results
print("Accuracy Score: {:.3f}".format(SVM_Accuracy_Score))
print("Jaccard Index: {:.3f}".format(SVM_JaccardIndex))
print("F1 Score: {:.3f}".format(SVM_F1_Score))
```

```
Accuracy Score: 0.834
Jaccard Index: 0.495
F1 Score: 0.826
```

Report

Q19) 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 [147]:

```
# creating a dictionary with the values for each metric
metrics = {'KNN': [KNN_Accuracy_Score, KNN_JaccardIndex, KNN_F1_Score, 'N/A'],
           'Decision Tree': [Tree_Accuracy_Score, Tree_JaccardIndex, Tree_F1_Score,
           'Logistic Regression': [LR_Accuracy_Score, LR_JaccardIndex, LR_F1_Score,
           'SVM': [SVM_Accuracy_Score, SVM_JaccardIndex, SVM_F1_Score, 'N/A']}

# creating the data frame
metrics_df = pd.DataFrame.from_dict(metrics, orient='index', columns=['Accuracy', 'J

# the data frame
print(metrics_df)
```

```
                     Accuracy   Jaccard  F1-Score   LogLoss
KNN                  0.818321  0.687588  0.802375       N/A
Decision Tree        0.818321  0.697011  0.813263       N/A
Logistic Regression  0.835115  0.504587  0.670732  0.377798
SVM                  0.833588  0.495370  0.826482       N/A
```

How to submit

Once you complete your notebook you will have to share it. You can download the notebook by navigating to "File" and clicking on "Download" button.

This will save the (.ipynb) file on your computer. Once saved, you can upload this file in the "My Submission" tab, of the "Peer-graded Assignment" section.

About the Authors:

[Joseph Santarcangelo (https://www.linkedin.com/in/joseph-s-50398b136/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NSkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkML0101ENSkillsNetwork20718538-2022-01-01)] has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Other Contributors

[Svitlana Kramar (https://www.linkedin.com/in/svitlana-kramar/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NSkillsNetwork-Channel-SkillsNetworkCoursesIBMML0232ENSkillsNetwork30654641-2022-01-01)]

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-06-22	2.0	Svitlana K.	Deleted GridSearch and Mock