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/.

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	obje

Field	Description	Unit	Type
			ct
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
WindSpeed 9am	Wind speed averaged of 10 minutes prior to 9am	Kilometers/ Hour	float
WindSpeed 3pm	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

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

```
# All Libraries required for this lab are listed below. The libraries pre-installed on Skills Network Labs are commented.
# !mamba install -qy pandas==1.3.4 numpy==1.21.4 seaborn==0.9.0 matplotlib==3.5.0 scikit-learn==0.20.1
# Note: If your environment doesn't support "!mamba install", use "!pip install"

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

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

```
df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillUp/labs/ML-FinalAssignment/Weather_Data.csv')
df.head()

       Date  MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  
WindGustDir  \
0  2/1/2008     19.5     22.4      15.6          6.2       0.0   
W
```

```
1  2/2/2008     19.5     25.6        6.0          3.4       2.7
W
2  2/3/2008     21.6     24.5        6.6          2.4       0.1
W
3  2/4/2008     20.2     22.8       18.8          2.2       0.0
W
4  2/5/2008     19.7     25.7       77.4          4.8       0.0
W

   WindGustSpeed WindDir9am WindDir3pm  ...  Humidity9am  Humidity3pm
\
0             41          S        SSW  ...           92           84

1             41          W          E  ...           83           73

2             41        ESE        ESE  ...           88           86

3             41        NNE          E  ...           83           90

4             41        NNE          W  ...           88           74


   Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm
RainToday  \
0       1017.6       1017.4         8         8     20.7     20.9
Yes
1       1017.9       1016.4         7         7     22.4     24.8
Yes
2       1016.7       1015.6         7         8     23.5     23.0
Yes
3       1014.2       1011.8         8         8     21.4     20.9
Yes
4       1008.3       1004.8         8         8     22.5     25.5
Yes

   RainTomorrow
0           Yes
1           Yes
2           Yes
3           Yes
4           Yes

[5 rows x 22 columns]
```

Note: This version of the lab is designed for JupyterLite, which necessitates downloading the dataset to the interface. However, when working with the downloaded version of this notebook on your local machines (Jupyter Anaconda), you can simply **skip the steps above of "Importing the Dataset"** and use the URL directly in the `pandas.read_csv()` function. You can uncomment and run the statements in the cell below.

```
#filepath = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillUp/labs/ML-FinalAssignment/Weather_Data.csv"
#df = pd.read_csv(filepath)
```

Data Preprocessing

One Hot Encoding

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

```
df_sydney_processed = pd.get_dummies(data=df, columns=['RainToday', 'WindGustDir', 'WindDir9am', 'WindDir3pm'])
```

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.

```
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.

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

df_sydney_processed = df_sydney_processed.astype(float)

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`.

```
#Enter Your Code and Execute
X_train, X_test, y_train, y_test = train_test_split(features, Y, test_size=0.2, random_state=10)
```

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

```
#Enter Your Code and Execute
```

```
from sklearn import linear_model
regr = linear_model.LinearRegression()
LinearReg = regr.fit(X_train,y_train)
```

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

```
#Enter Your Code and Execute

predictions = regr.predict(X_test)
```

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

```
#Enter Your Code and Execute

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

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

```
#Enter Your Code and Execute

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

	Metric	Value
0	MAE	0.256318
1	MSE	0.115721
2	R2	0.427132

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`.

```
from sklearn.neighbors import KNeighborsClassifier

KNN = KNeighborsClassifier(n_neighbors = 4).fit(X_train,y_train)
KNN

KNeighborsClassifier(n_neighbors=4)
```

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

```
#Enter Your Code and Execute

predictions = KNN.predict(X_test)
predictions[0:5]

array([0., 0., 1., 0., 0.])
```

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

```
#Enter Your Code and Execute

KNN_Accuracy_Score = metrics.accuracy_score(y_train,
KNN.predict(X_train))
KNN_JaccardIndex =   metrics.accuracy_score(y_test, predictions)
KNN_F1_Score = metrics.f1_score(y_test, predictions)
Report = pd.DataFrame({
    'Metric': ['KNN_Accuracy_Score', 'KNN_JaccardIndex',
'KNN_F1_Score'],
    'Value': [KNN_Accuracy_Score, KNN_JaccardIndex, KNN_F1_Score]
})
Report
```

	Metric	Value
0	KNN_Accuracy_Score	0.858180
1	KNN_JaccardIndex	0.818321
2	KNN_F1_Score	0.596610

Decision Tree

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

```
#Enter Your Code and Execute

Tree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
Tree # it shows the default parameters
Tree.fit(X_train,y_train)

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`.

```
#Enter Your Code and Execute

predictions = Tree.predict(X_test)
predictions[0:5]
```

```
array([0., 0., 1., 0., 0.])
```

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

```
#Enter Your Code and Execute

Tree_Accuracy_Score = metrics.accuracy_score(y_train,
Tree.predict(X_train))
Tree_JaccardIndex = metrics.accuracy_score(y_test, predictions)
Tree_F1_Score = metrics.f1_score(y_test, predictions)
Report = pd.DataFrame({
    'Metric': ['Tree_Accuracy_Score', 'Tree_JaccardIndex',
'Tree_F1_Score'],
    'Value': [Tree_Accuracy_Score, Tree_JaccardIndex, Tree_F1_Score]
})
Report
```

	Metric	Value
0	Tree_Accuracy_Score	0.746942
1	Tree_JaccardIndex	0.601527
2	Tree_F1_Score	0.225519

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`.

```
#Enter Your Code and Execute

x_train, x_test, y_train, y_test = train_test_split(features, Y,
test_size=0.2, random_state=1)
```

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

```
#Enter Your Code and Execute

LR = LogisticRegression(C=0.01,
solver='liblinear').fit(X_train,y_train)
```

Q14) Now, use the `predict` and `predict_proba` methods on the testing data (`x_test`) and save it as 2 arrays `predictions` and `predict_proba`.

```
#Enter Your Code and Execute

predictions = LR.predict(X_test)

predict_proba = LR.predict_proba(X_test)
```

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

```
#Enter Your Code and Execute

from sklearn.metrics import log_loss
LR_Accuracy_Score = metrics.accuracy_score(y_train,
Tree.predict(X_train))
LR_JaccardIndex = metrics.jaccard_score(y_test,
predictions,pos_label=0)
LR_F1_Score = metrics.f1_score(y_test, predictions)
LR_Log_Loss = metrics.log_loss(y_test,predict_proba)
```

SVM

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

```
#Enter Your Code and Execute

x_train, x_test, y_train, y_test = train_test_split(features, Y,
test_size=0.2, random_state=10)
SVM = svm.SVC(kernel='rbf')
SVM.fit(X_train, y_train)

SVC()
```

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

```
#Enter Your Code and Execute

predictions = SVM.predict(X_test)
```

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

```

SVM_Accuracy_Score = metrics.accuracy_score(y_train,
SVM.predict(X_train))
SVM_JaccardIndex = metrics.f1_score(y_test, predictions)
SVM_F1_Score = metrics.f1_score(y_test, predictions)
Report = pd.DataFrame({
    'Metric': ['SVM_Accuracy_Score', 'SVM_JaccardIndex',
'SVM_F1_Score'],
    'Value': [SVM_Accuracy_Score, SVM_JaccardIndex, SVM_F1_Score]
})
Report
```

	Metric	Value
0	SVM_Accuracy_Score	0.745795

```
1    SVM_JaccardIndex  0.000000
2        SVM_F1_Score  0.000000
```

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

```
data = np.array([[KNN_Accuracy_Score , KNN_JaccardIndex,
KNN_F1_Score,''],
                 [Tree_Accuracy_Score, Tree_JaccardIndex,
Tree_F1_Score,''],
                 [LR_Accuracy_Score, LR_JaccardIndex,
LR_F1_Score,LR_Log_Loss],

[SVM_Accuracy_Score,SVM_JaccardIndex,SVM_F1_Score,'']])

row_headers = ['KNN', 'Decision Tree', 'Logistic Regression','SVM']
column_headers = ['Accuracy_Score', 'JaccardIndex',
'F1_Score','Log_Loss']
Report = pd.DataFrame(data, index=row_headers, columns=column_headers)
Report
```

	Accuracy_Score	JaccardIndex	\
KNN	0.8581804281345565	0.8183206106870229	
Decision Tree	0.7469418960244648	0.601526717557252	
Logistic Regression	0.7469418960244648	0.5769854132901134	
SVM	0.7457951070336392	0.0	

	F1_Score	Log_Loss
KNN	0.5966101694915255	
Decision Tree	0.22551928783382788	
Logistic Regression	0.22551928783382788	0.5971474614603932
SVM	0.0	

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.

Joseph Santarcangelo 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

© IBM Corporation 2020. All rights reserved.

<!--

Change Log

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

--!>