2020/2021 First Semester

|  |  |
| --- | --- |
| Course Code | DS540 |
| Course Name | Advanced Python for Data Science |
| CRN | 14045 |
| Assignment type | Critical Thinking |
| Module | 12 |
| Assignment Points | 100 |

|  |  |
| --- | --- |
| Student ID | G200007195 |
| Student Name | Mousa Ageeli |

**Machine Learning classification models to Iris flowers dataset**

**The project tasks:**

1. Import libraries

import sys

import traceback

import numpy as np

import pandas as pd

from sklearn import preprocessing, metrics

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report, confusion\_matrix

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KneighborsClassifier

1. Data Loading

iris = pd.read\_csv('/Users/ogail/Downloads/iris.csv')

1. Exploratory Data Analysis

print(f"First 6 Records\n {iris.head()}")

# get Information about data

print(f"info {iris.info()}")

print(f"Value count for each type :\n{iris['variety'].value\_counts()}")

Output

First 6 Records

sepal.length sepal.width petal.length petal.width variety

0 5.1 3.5 1.4 0.2 Setosa

1 4.9 3.0 1.4 0.2 Setosa

2 4.7 3.2 1.3 0.2 Setosa

3 4.6 3.1 1.5 0.2 Setosa

4 5.0 3.6 1.4 0.2 Setosa

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 150 entries, 0 to 149

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 sepal.length 150 non-null float64

1 sepal.width 150 non-null float64

2 petal.length 150 non-null float64

3 petal.width 150 non-null float64

4 variety 150 non-null object

dtypes: float64(4), object(1)

memory usage: 6.0+ KB

info None

Value count for each type :

Virginica 50

Versicolor 50

Setosa 50

Name: variety, dtype: int64

1. visualization data

sns.pairplot(iris, hue='variety', markers='+')

plt.show()

Output

A picture containing building, window

Description automatically generated

1. Data Cleaning(Preprocessing)

# label\_encoder object knows how to understand word labels.

label\_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'variety'.

ris['variety'] = label\_encoder.fit\_transform(iris['variety'])

# result after encoding

print(f"Result after encoding:\n{iris['variety'].unique()}")

# Modeling with scikit-learn

# Model Selection and Creation

X = iris.drop(['variety'], axis=1)

y = iris['variety']

# Data Splitting [Test,Training]

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=5)

print(f"Shape of X-Training : {x\_train.shape}")

print(f"Shape of Y-Training : {y\_train.shape}")

print(f"Shape of X-Test : {x\_test.shape}")

print(f"Shape of Y-Test : {y\_test.shape}")

Output

Result after encoding:

[0 1 2]

Shape of X-Training : (90, 4)

Shape of Y-Training : (90,)

Shape of X-Test : (60, 4)

Shape of Y-Test : (60,)

Length of list 25

Max of list 1.0

1. K Nearest Neighbors and Random Forests

#range of k we want to try

k\_range = list(range(1, 26))

# empty list to store scores

scores = []

# Model Selection and Creation

# loop through reasonable values of k

for k in k\_range:

# run KNeighborsClassifier with k neighbours

knn = KNeighborsClassifier(n\_neighbors=k)

# Train the model

knn.fit(x\_train, y\_train)

y\_pred = knn.predict(x\_test)

# append scores for k neighbors to scores list

scores.append(metrics.accuracy\_score(y\_test, y\_pred))

# we should have 26 scores here

print(f'Length of list {len(scores)}')

print(f'Max of list {max(scores)}')

# plot how accuracy changes as we vary k

plt.plot(k\_range, scores)

plt.xlabel('Value of k for KNN')

plt.ylabel('Accuracy Score')

plt.title('Accuracy Scores for Values of k of k-Nearest-Neighbors')

plt.show()

output

A picture containing line chart

Description automatically generated

1. Calculate the following classification metrics to validate the model

log\_reg = LogisticRegression()

log\_reg.fit(x\_train, y\_train)

y\_pred = log\_reg.predict(x\_test)

# Summary of the predictions made by the classifier

print\_report(y\_pred, y\_test)

# Create a Gaussian Classifier

clf = RandomForestClassifier(n\_estimators=100)

# Train the model

clf.fit(x\_train, y\_train)

rf\_pred = clf.predict(x\_test)

# Summary of the predictions made by the classifier

print\_report(rf\_pred, y\_test)

Output

Classification Report

precision recall f1-score support

0 1.00 1.00 1.00 20

1 1.00 0.95 0.98 21

2 0.95 1.00 0.97 19

accuracy 0.98 60

macro avg 0.98 0.98 0.98 60

weighted avg 0.98 0.98 0.98 60

Confusion Matrix

[[20 0 0]

[ 0 20 1]

[ 0 0 19]]

Accuracy Score0.9833333333333333

Classification Report

precision recall f1-score support

0 1.00 1.00 1.00 20

1 0.91 0.95 0.93 21

2 0.94 0.89 0.92 19

accuracy 0.95 60

macro avg 0.95 0.95 0.95 60

weighted avg 0.95 0.95 0.95 60

Confusion Matrix

[[20 0 0]

[ 0 20 1]

[ 0 2 17]]

Accuracy Score0.95