3803ICT

Big Data Analysis

Lab 04 – Data Analysis and Interpretation - Statistical Data Analysis

Trimester 1 - 2019

Table of Contents

I. Classification

In this exercise, you'll be working with the MNIST digits recognition dataset, which has 10 classes, the digits 0 through 9! A reduced version of the MNIST dataset is one of scikit-learn's included datasets, and that is the one we will use in this exercise.

Each sample in this scikit-learn dataset is an 8x8 image representing a handwritten digit. Each pixel is represented by an integer in the range 0 to 16, indicating varying levels of black.

To load dataset, using the following code:

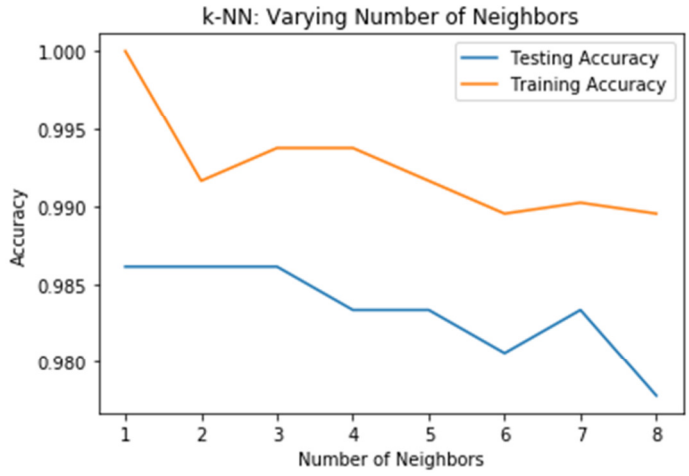
```
# Import necessary modules
from sklearn import datasets
import matplotlib.pyplot as plt

# Load the digits dataset: digits
digits = datasets.load_digits()
```

Before applying the classifier, we need to split the dataset:

```
from sklearn.model_selection import train_test_split
import numpy as np

# Create feature and target arrays
X = digits.data
y = digits.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
```

1. k-Nearest Neighbors

- Implement kNN classification for the above dataset.
- Compute and plot the accuracy scores by k values.



- Make conclusions about which k is the best.

2. Support Vector Machine

- Implement SVM classifier for MNIST dataset.

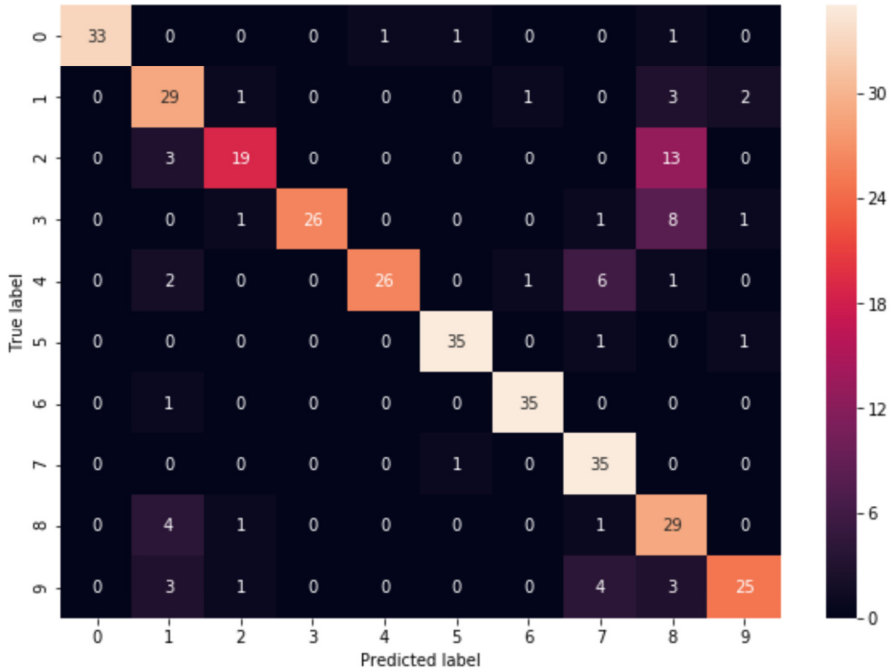
- Compute and compare the accuracy scores for at least 3 different kernels by using metrics.classification_report.

```
Classification report for classifier SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        36
           1       0.92      0.94      0.93        36
           2       1.00      1.00      1.00        35
           3       1.00      0.97      0.99        37
           4       1.00      1.00      1.00        36
           5       1.00      1.00      1.00        37
           6       1.00      0.97      0.99        36
           7       0.97      1.00      0.99        36
           8       0.94      0.89      0.91        35
           9       0.95      1.00      0.97        36

   micro avg       0.98      0.98      0.98       360
   macro avg       0.98      0.98      0.98       360
weighted avg       0.98      0.98      0.98       360
```

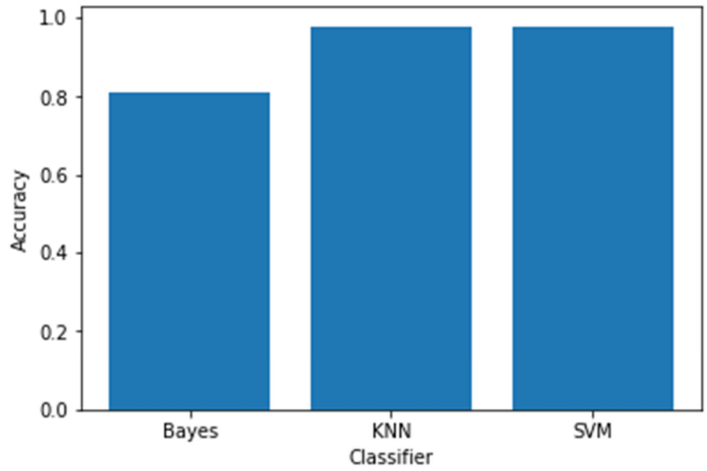
- Make conclusion about which kernel should be used in this case.

3. Naïve Bayes

- Compute the accuracy scores and plot the confusion matrix of the same dataset using Naïve Bayes classifier. Hint: from sklearn.metrics import confusion_matrix



- Compare the accuracy of different classifiers in the same plot.

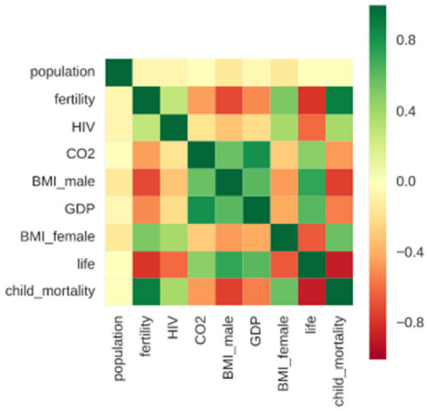


II. Regression

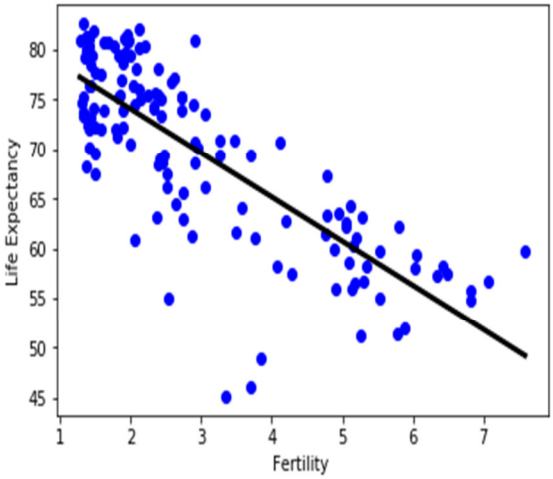
1. Linear Regression

You will work with Gapminder data that in CSV file available in the workspace as 'gapminder.csv'. Specifically, your goal will be to use this data to predict the life expectancy in a given country based on features such as the country's GDP, fertility rate, and population.

- ❖ Use seaborn to visualize the data of Gapminder like following image:



- ❖ Apply linear regression with the 'fertility' feature to predict life expectancy.



- Apply linear regression with the **all** features to predict life expectancy. *Compare the model score when using all features to one feature in previous step.*
- *Apply 5-fold cross-validation (for both 2 above steps) and compare your model score accuracy. Hint: from sklearn.model_selection import cross_val_score.*

2. Logistic Regression

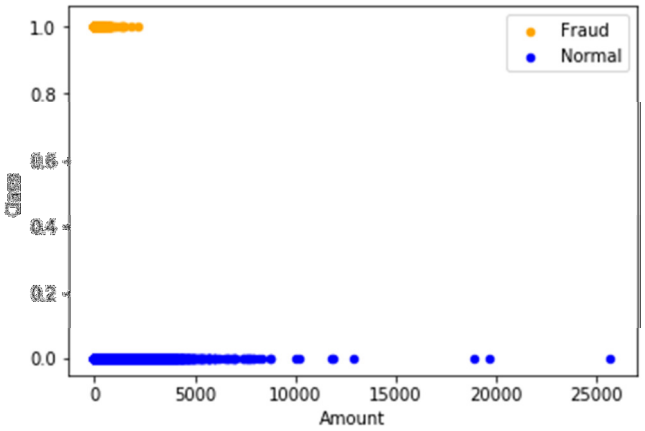
We will be detecting credit card fraud based on the different features of our dataset with Logistic Regression.

- Load the 'creditcard.csv' file into dataframe. The attribute 'Class' is marked for fraud transactions (Class = 1).

```
frauds = df.loc[df['Class'] == 1]
non_frauds = df.loc[df['Class'] == 0]
print(len(frauds), "frauds, ", len(non_frauds), "nonfrauds.")
```

```
492 frauds,  284315 nonfrauds.
```

- Compare the relation between Class and Amount. Conclusion?



- Use Logistic Regression to predict the fraud transactions and plot the confusion matrix of the model.