Practical - 2

2CS702 – Big Data Analytics

Harshit Gajipara
19BCE059

Practical 2

Learning limitation of data analytics by applying Machine Learning Techniques on large amount of data. Write a program to read data set from any online website, excel file and CSV file and to perform a) Linear regression and logistic regression on iris dataset. b) K-means clustering.

Importing libraries

```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
```

Linear Regression

```
# Load Iris Data
iris = load_iris()

# Creating pd DataFrames
iris_df = pd.DataFrame(data= iris.data, columns= iris.feature_names)
target_df = pd.DataFrame(data= iris.target, columns= ['species'])

def converter(specie):
    if specie == 0:
        return 'setosa'
    elif specie == 1:
        return 'versicolor'
    else:
        return 'virginica'

target_df['species'] = target_df['species'].apply(converter)

# Concatenate the DataFrames
iris_df = pd.concat([iris_df, target_df], axis= 1)

# Converting Objects to Numerical dtype
iris_df.drop('species', axis= 1, inplace= True)
target_df = pd.DataFrame(columns= ['species'], data= iris.target)
iris_df = pd.concat([iris_df, target_df], axis= 1)

X= iris_df.drop(labels= 'sepal length (cm)', axis= 1)
y= iris_df['sepal length (cm)']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
0.33, random_state= 101)

lr = LinearRegression()
lr.fit(X_train, y_train)
pred = lr.predict(X_test)

# Evaluating Model's Performance
print('Mean Absolute Error:', mean_absolute_error(y_test, pred))
print('Mean Squared Error:', mean_squared_error(y_test, pred))
print('Mean Root Squared Error:', np.sqrt(mean_squared_error(y_test,
pred)))
```

```
Mean Absolute Error: 0.2595570975563036
Mean Squared Error: 0.10174529564238954
Mean Root Squared Error: 0.3189753840696638
```

```
d = {'sepal length (cm)' : [4.6],
     'sepal width (cm)' : [3.4],
     'petal length (cm)' : [1.4],
     'petal width (cm)' : [0.3],
     'species' : 0}

test_df = pd.DataFrame(data= d)
pred = lr.predict(X_test)

print('Predicted Sepal Length (cm):', pred[0])
print('Actual Sepal Length (cm):', 4.6)
```

```
Predicted Sepal Length (cm): 5.461145872156033
Actual Sepal Length (cm): 4.6
```

Logistic Regression

```
data = load_iris()
x_train, x_test, y_train, y_test = train_test_split(data.data,
data.target, test_size=0.2)

reg = LogisticRegression()
reg.fit(x_train,y_train)

print(reg.predict(x_test),y_test)
print(reg.score(x_test,y_test))
```

```
[0 2 1 0 2 0 0 1 0 1 0 0 0 1 1 1 1 2 0 1 0 2 0 1 2 2 2 1 1 0] [0 2 1 0
2 0 0 1 0 1 0 0 0 1 1 1 1 2 0 1 0 2 0 1 2 2 2 1 1 0]
1.0

C:\Users\HARSHIT\anaconda3\lib\site-packages\sklearn\linear_model\
_logistic.py:814: ConvergenceWarning: lbfgs failed to converge
(status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

K-means clustering

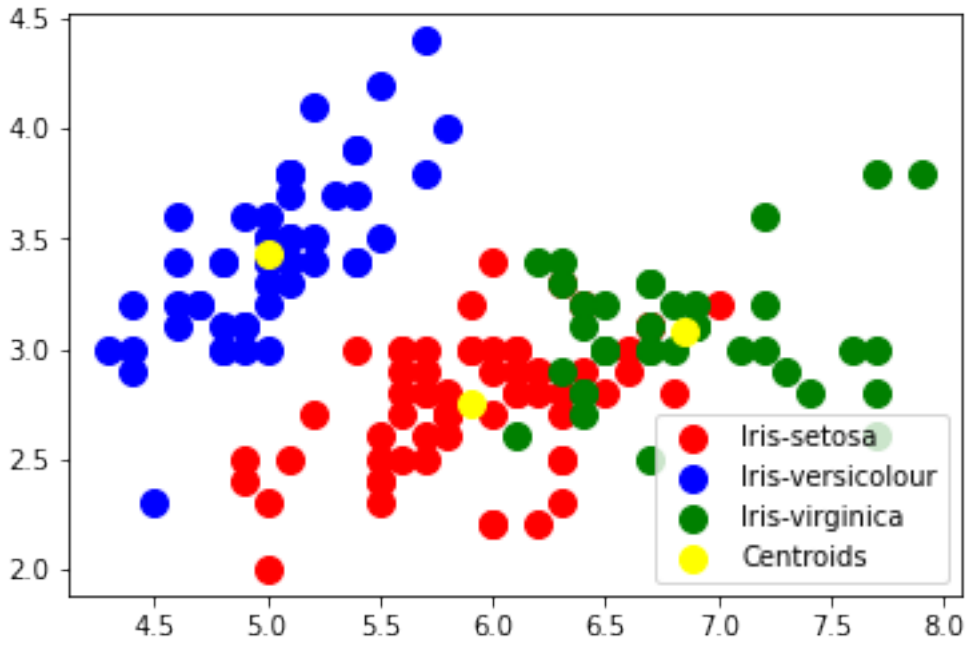
```
data = pd.DataFrame(load_iris()['data'])
x = data.iloc[:, [0, 1, 2, 3]].values

kmeans = KMeans(n_clusters = 3, max_iter = 300, n_init = 10,
random_state = 0)
y_kmeans = kmeans.fit_predict(x)

#Visualising the clusters
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c =
'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c =
'blue', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c =
'green', label = 'Iris-virginica')

#Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 0],
kmeans.cluster_centers_[:,1], s = 100, c = 'yellow', label =
'Centroids')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1ababb12a60>
```


4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Iris-setosa
Iris-versicolour
Iris-virginica
Centroids