

INFO 4330

Data Warehouse/Mining Final Project

James Tariga
Ali Manghat



Table of contents

01.

Background

02.

Linear
Regression

03.

Clustering

04.

Comparing Results

05.

Logistic
Regression/K-
Nearest Neighbors

06.

Summary

Background

- Explored COVID-19 data using various machine learning models
- The chosen dataset provided a detailed overview of the recent pandemic and COVID-19 statistics for various regions worldwide
- Applied supervised learning models including Linear Regression, K-Nearest Neighbors, and Logistical Regression
- Investigated the topic of unsupervised learning through Clustering.
- Applied many of the machine learning techniques learned in class to this real-world dataset

Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered
Afghanistan	36263	1269	25198	9796	106	10	18	3.5	69.49	5.04
Albania	4880	144	2745	1991	117	6	63	2.95	56.25	5.25

Linear Regression

- Used 'Confirmed', 'Active', 'Recovered / 100 Cases' as the independent variables, and 'Death / 100 cases' as the dependent variable.
- Split the data set for the first 131 rows for training and the remaining for testing. There are a total of 188 rows.
- Created a linear regression model.
- Trained the model using the training data. We make predictions on the model using the testing data.

```
1172] df = pd.read_csv('country_wise_latest.csv')

1173] x = df[['Confirmed', 'Active', 'Recovered / 100 Cases']]
      y = df['Deaths / 100 Cases']

      x_train = x[:131]
      x_test = x[131:]

      y_train = y[:131]
      y_test = y[131:]

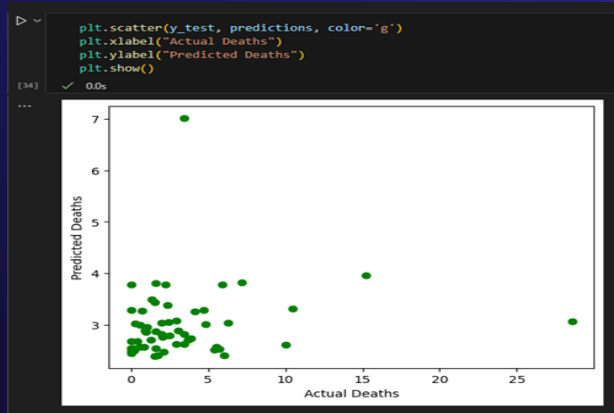
1174]
```

```
linearRegModel = LinearRegression()
linearRegModel.fit(x_train, y_train)
[5] ✓ 0.0s
...
* LinearRegression
  LinearRegression()

▷ y predictions = linearRegModel.predict(x_test)
[6] ✓ 0.0s
```

Linear Regression

- The scatter plot shows the actual and predicted values of the deaths / 100 cases.
- The MSE is 19.78 which is relatively low. This means that the model is pretty accurate. The predicted values are like the actual values. We can see the max value for the dependent variable 'Deaths / 100 cases' is 28.56 and the lowest is 0. This further proves that the MSE is relatively accurate.



```
mse = mean_squared_error(y_test, predictions)
print(mse)
```

```
35] ✓ 0.0s
.. 19.77795136072621
```

```
max_value = df['Deaths / 100 Cases'].max()
print("Max Value: ", max_value)
```

```
min_value = df['Deaths / 100 Cases'].min()
print("Min Value: ", min_value)
```

```
36] ✓ 0.0s
.. Max Value: 28.56
   Min Value: 0.0
```

Clustering

- There two clustering models. The first with 2 clusters, and the second with 3 clusters.

```
km1 = KMeans(n_clusters=2, n_init=20)
km1.fit(x_train)
```

[1180]

...

▼ KMeans

KMeans(n_clusters=2, n_init=20)

```
km2 = KMeans(n_clusters=3, n_init=20)
km2.fit(x_train)
```

[1181]

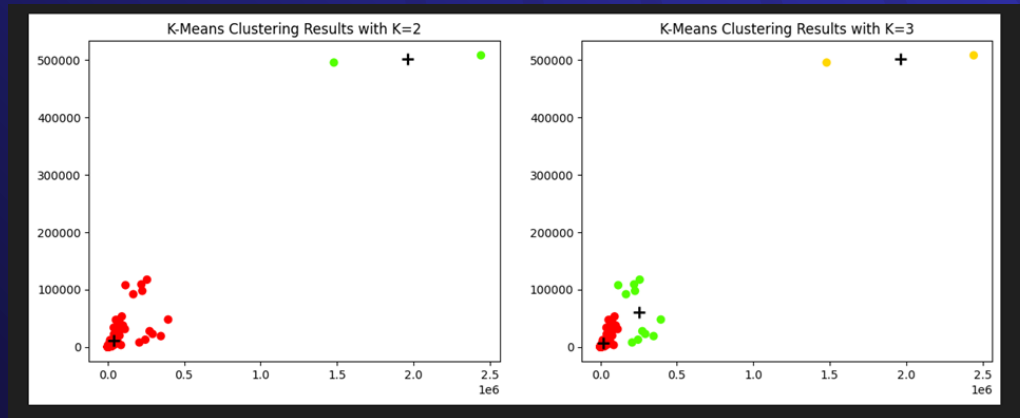
...

▼ KMeans

KMeans(n_clusters=3, n_init=20)





Clustering

- Plotting the clustered data. Here we can see the two models plotted.
- The graph on the left represents the first model which is grouping the data into two clusters and on the right the model is grouping the data in three clusters.
- We can see covid-19 cases/deaths grouped by different regions.
- First world countries have a much lower number of deaths, the number increases as the countries become less developed. The centroids have also been plotted to show the 'mean' for each of the clusters.





Comparing Results

- Both Linear Regression and Clustering have specific use cases.
 - The choice of model depends on the prediction goal.
 - In our scenario, Linear Regression is used to predict the number of deaths based on features like confirmed cases, recovered cases, and active cases.
 - Clustering helps us visualize regions with the highest death rates but doesn't help us predict deaths based on the features.
 - Linear Regression is more appropriate for this scenario for prediction purposes.
 - Clustering is useful for visualizing similarities in the data.
- 
- 
- 
- 

Logistic Regression/K-Nearest Neighbors

- Feature is xKNN and a dependent called yKNN to train the model. Then the models are trained using the fit method on the provided training data.
- We then make predictions on the same training data using the trained models.

```
xKNN = df[['Confirmed', 'Active', 'Recovered / 100 Cases', 'Deaths / 100 Cases']]
yKNN = df['Americas']

logregmodel = LogisticRegression()
knn = KNeighborsClassifier(5)

logregmodel.fit(xKNN, yKNN)
knn.fit(xKNN, yKNN)

KNeighborsClassifier
KNeighborsClassifier()
```

```
logmodelpreds = logregmodel.predict(xKNN)
knnmodelpreds = knn.predict(xKNN)
```

Logistic Regression/K-Nearest Neighbors

- Confusion matrices are computed for both models using the true labels (yKNN) and predicted labels (logmodelpreds, knnmodelpreds).
- The accuracy scores are calculated for both models using the true labels and predicted labels.

```
logconf = confusion_matrix(yKNN, logmodelpreds, labels=knn.classes_)
print("Logistical Regression: \n", logconf)

KNNconf = confusion_matrix(yKNN, knnmodelpreds, labels=knn.classes_)
print("KNN: \n", KNNconf)
```

✓ 0.0s

Logistical Regression:

```
[[148  4]
 [ 32  3]]
```

KNN:

```
[[148  4]
 [ 27  8]]
```

```
logacc = accuracy_score(yKNN, logmodelpreds)
print("Logistical Regression: ", logacc)
```

```
KNNacc = accuracy_score(yKNN, knnmodelpreds)
print("KNN: ", KNNacc)
```


✓ 0.0s

... Logistical Regression: 0.8074866310160428
KNN: 0.8342245989304813




Logistic Regression/K-Nearest Neighbors

- Precision, recall, and F1-score are computed using the `precision_recall_fscore_support` function for both models, considering a binary classification scenario.
- This essentially trains two classifiers, makes predictions, evaluates their performance using confusion matrices, accuracy scores, and precision-recall-f1 metrics. It provides a comprehensive overview of how well each model is performing on the given dataset.



```
precision_recall_fscore_support(yKNN, logmodelpreds, average='binary')
127] ✓ 0.0s
```

```
.. (0.42857142857142855, 0.08571428571428572, 0.14285714285714285, None)
```







```
precision_recall_fscore_support(yKNN, knnmodelpreds, average='binary')
128] ✓ 0.0s
```

```
.. (0.6666666666666666, 0.22857142857142856, 0.3404255319148936, None)
```



Comparing Results

- Logistic Regression model performs well on majority class but struggles with minority class.
 - KNN model shows slight improvement over Logistic Regression in predicting minority class.
 - KNN has better overall prediction accuracy.
 - Both models struggle with minority class due to imbalanced dataset.
 - Improvements can be made by resampling dataset, trying different algorithms, feature selection methods, adjusting decision threshold, and addressing class imbalance.
- 
- 
- 
- 



Reference

Citation of data set:

Devakumar, K.P. (n.d.). Corona Virus Report. Kaggle. Retrieved from <https://www.kaggle.com/datasets/imdevskp/corona-virus-report>

