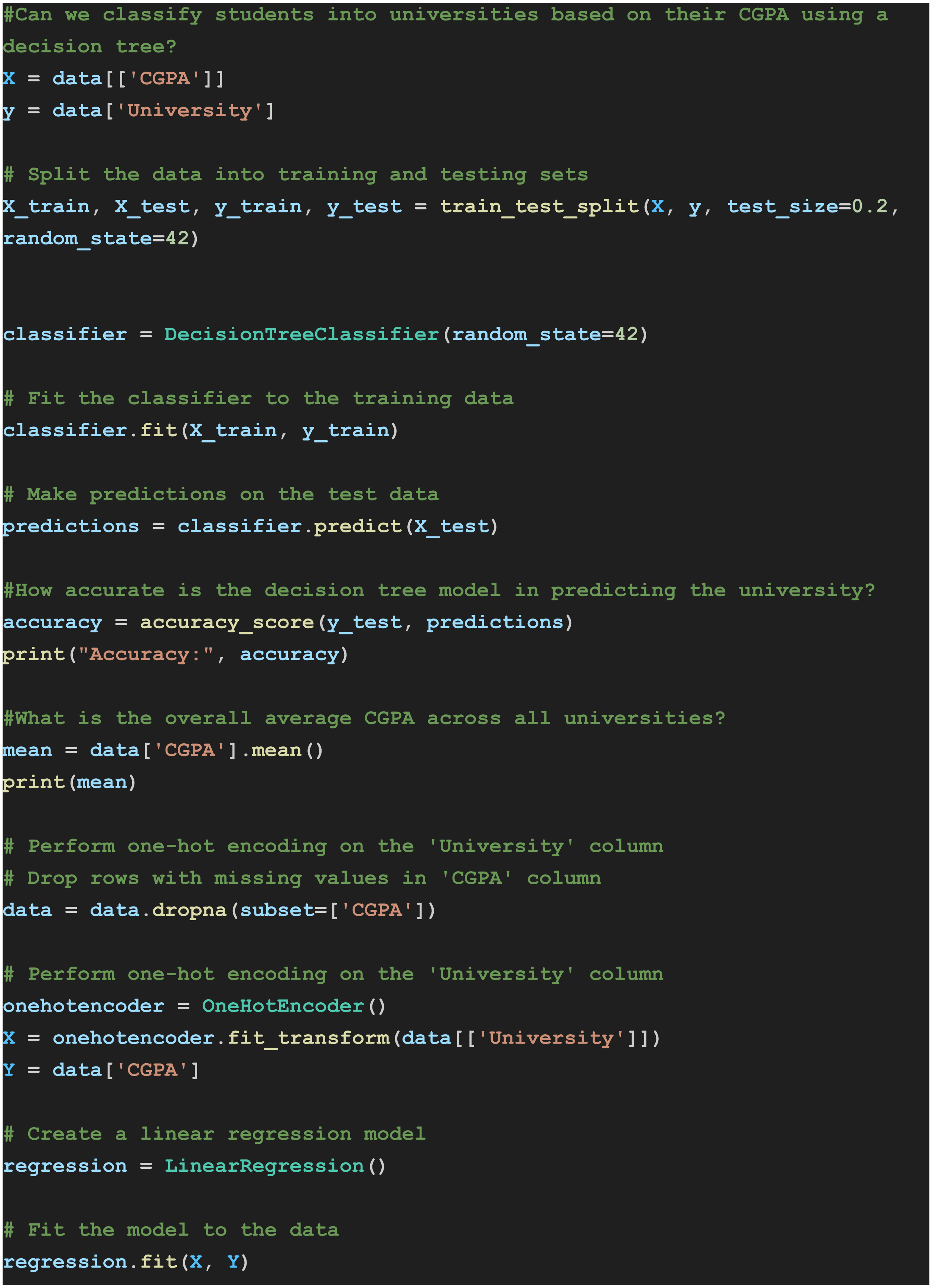
```python
# -*- coding: utf-8 -*-
"""Alaiba_Nawaz_Day5_W2.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1A5E56fjtBC-j1Nckqzkilm1m2xStVjC_
"""

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression , LogisticRegression
from scipy import stats
import warnings
warnings.filterwarnings("ignore")
from sklearn.cluster import KMeans , AgglomerativeClustering
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score , mean_squared_error
from sklearn.preprocessing import OneHotEncoder

data = pd.read_csv("/content/students_data.csv")

data.info()

data.head(5)

#naming columns
data = pd.read_csv("/content/students_data.csv" , names = ['Name' ,
'University' , 'CGPA' , 'NaN'])

data.head(5)

#converting cgpa to numeric
data['CGPA'].unique()
```

```
#removing all non numeric data
data['CGPA'] = pd.to_numeric(data['CGPA'], errors='coerce')
data['CGPA'].unique()

#dropping all nulls
data.dropna()

#dropping nan column
data.drop('NaN' , axis = 1)

#What are the top 5 universities with the highest average CGPA?
result =
data.groupby('University')['CGPA'].mean().sort_values(ascending=False).hea
d(5)
print(result)

#Is there a correlation between the CGPA and the length of the student's
name?
correlation = data['CGPA'].corr(data['Name'].str.len())
print(correlation)

#How does the CGPA vary across different universities?
plt.figure(figsize=(10, 8))
sns.boxplot(data=data, x='University', y='CGPA')
plt.title('CGPA Variation Across Universities')
plt.xlabel('University')
plt.xticks(rotation=90)
plt.ylabel('CGPA')
plt.show()

#Can we predict a student's CGPA based on the length of their name using
linear regression?
data['Name_Length'] = data['Name'].apply(len)
x = data['Name_Length']
y = data['CGPA']
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=42)

# Create a linear regression model
model = LinearRegression()
```

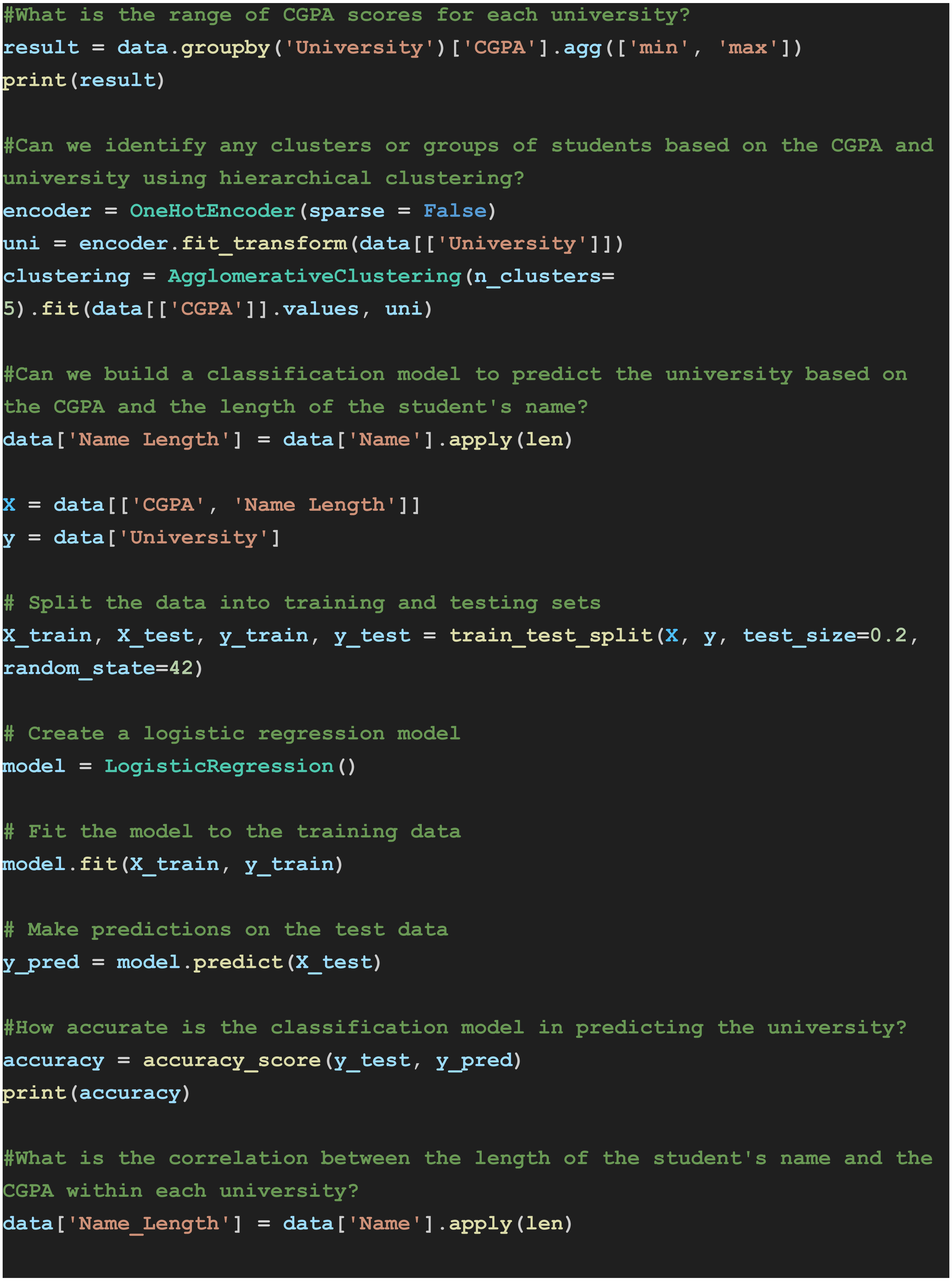
```
X_train = X_train.values.reshape(-1, 1)
y_train = y_train.values.reshape(-1, 1)
X_test = X_test.values.reshape(-1,1)
# Fit the model to the training data
model.fit(X_train.reshape(-1, 1), y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Plot the data and the linear regression line
plt.scatter(x, y, label='Data')
plt.plot(X_test, y_pred, color='red', label='Linear Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()


#Which university has the highest number of students with a CGPA above a
certain threshold?
threshold = 2.5
above_threshold  = data[data['CGPA'] >= threshold]
result =
above_threshold.groupby('University')['Name'].count().sort_values(ascending=False).head(1)
print(result)

#Can we identify any outliers in the CGPA distribution within each
university?

# Group the data by 'University'
grouped_data = data.groupby('University')['CGPA']

# Calculate mean and standard deviation for each university
mean_cgpas = grouped_data.mean()
std_cgpas = grouped_data.std()

# Compute z-scores for each CGPA value within each university
z_scores = grouped_data.apply(lambda x: (x - x.mean()) / x.std())
```

```python
# Define a threshold for identifying outliers (e.g., z-score greater than 2 or -2)
threshold = 4

# Identify outliers for each university
outliers_by_university = data[z_scores.abs() > threshold]

print(outliers_by_university)


#Can we cluster students based on their CGPA using k-means clustering?
x = data['CGPA']
inertias = []
x = x.values.reshape(-1,1)
for i in range(1,11):
  kmeans = KMeans(n_clusters = i)
  kmeans.fit(x)
  inertias.append(kmeans.inertia_)
plt.plot(range(1,11) , inertias , marker = 'o')
plt.title("Elbow Method")
plt.xlabel("Number of cluster")
plt.ylabel("Inertia")
plt.show()
#here we can see sharp change at k = 2 so taking k(no of cluster as 2)

kmeans = KMeans(n_clusters = 2)
kmeans.fit(x)
plt.scatter(x , np.zeros(len(data)) ,c = kmeans.labels_)
plt.show()

#What is the average CGPA for each cluster identified in the previous question?
# Get the cluster assignments for each data point
data['Cluster'] = kmeans.labels_

# Calculate the average CGPA for each cluster
average_cgpa_by_cluster = data.groupby('Cluster')['CGPA'].mean()

print(average_cgpa_by_cluster)
```



```
#Can we classify students into universities based on their CGPA using a
decision tree?
X = data[['CGPA']]
y = data['University']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


classifier = DecisionTreeClassifier(random_state=42)

# Fit the classifier to the training data
classifier.fit(X_train, y_train)

# Make predictions on the test data
predictions = classifier.predict(X_test)

#How accurate is the decision tree model in predicting the university?
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

#What is the overall average CGPA across all universities?
mean = data['CGPA'].mean()
print(mean)

# Perform one-hot encoding on the 'University' column
# Drop rows with missing values in 'CGPA' column
data = data.dropna(subset=['CGPA'])

# Perform one-hot encoding on the 'University' column
onehotencoder = OneHotEncoder()
X = onehotencoder.fit_transform(data[['University']])
Y = data['CGPA']

# Create a linear regression model
regression = LinearRegression()

# Fit the model to the data
regression.fit(X, Y)
```

```
# Make predictions on the data
prediction = regression.predict(X)

# Plot the data and the linear regression line
plt.scatter(data['University'], data['CGPA'], label='Data')
plt.plot(data['University'], prediction, color='red', label='Linear
Regression')
plt.xlabel('University')
plt.ylabel('CGPA')
plt.legend()
plt.show()

# How well does the regression model perform in predicting the CGPA?
mse = mean_squared_error(Y, prediction)
print(mse)

#Are there any missing or erroneous values in the CGPA column?
nulls = data['CGPA'].isnull().sum()
print('Number of nulls are : ' , nulls)
#max cgpa
max = data['CGPA'].max()
print(max)
#cgpa cannot be above than 4 removing all values above 4
data = data[data['CGPA'] < 4]

#Can we detect any relationships between the length of the student's name
and their university using association rules?
# Add a new column for the length of the student's name
data['Name Length'] = data['Name'].apply(len)

# Plot a box plot to visualize the relationship between name length and
university
sns.boxplot(x='University', y='Name Length', data=data)
plt.xlabel('University')
plt.ylabel('Name Length')
plt.title('Relationship between Name Length and University')
plt.xticks(rotation = 90)
plt.show()
```



```
#What is the range of CGPA scores for each university?
result = data.groupby('University')['CGPA'].agg(['min', 'max'])
print(result)

#Can we identify any clusters or groups of students based on the CGPA and
university using hierarchical clustering?
encoder = OneHotEncoder(sparse = False)
uni = encoder.fit_transform(data[['University']])
clustering = AgglomerativeClustering(n_clusters=
5).fit(data[['CGPA']].values, uni)

#Can we build a classification model to predict the university based on
the CGPA and the length of the student's name?
data['Name Length'] = data['Name'].apply(len)

X = data[['CGPA', 'Name Length']]
y = data['University']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create a logistic regression model
model = LogisticRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

#How accurate is the classification model in predicting the university?
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)

#What is the correlation between the length of the student's name and the
CGPA within each university?
data['Name_Length'] = data['Name'].apply(len)
```

```
# Group the data by 'University' and calculate the correlation between
'Name_Length' and 'CGPA' within each group
correlations = data.groupby('University')['Name_Length',
'CGPA'].corr().iloc[0::2, -1]

# Print the correlation for each university
print(correlations)
```