AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department of Artificial Intelligence & Data Science

Lab Manual

Data Structures & Algorithms LABORATORY

Computer Laboratory I: Machine Learning
(417521)

Prepared by
Mrs. Sonali Nawale

BE AI& DS
SEM –VII
Academic year-2023-24

Sr. No	Title Of Experiment	CO	PO	PSO	Page No
1B	Feature Transformation: Apply LDA Algorithm on Iris Dataset and classify which species a given flower belongs to. Dataset Link: https://www.kaggle.com/datasets/uciml/iris				
2B	Regression Analysis: Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following: a. Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis b. Bivariate analysis: Linear and logistic regression modeling c. Multiple Regression analysis d. Also compare the results of the above analysis for the two data sets Dataset link: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database				
3B	Classification Analysis: Implement K-Nearest Neighbors' algorithm on social network ad dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset. Dataset link: https://www.kaggle.com/datasets/rakeshrau/social-network-ads				
4A	Clustering Analysis: Implement K-Means clustering on Iris.csv dataset. Determine the number of clusters using the elbow method. Dataset Link: https://www.kaggle.com/datasets/uciml/iris				
5A	Ensemble Learning: Implement Random Forest Classifier model to predict the safety of the car. Dataset link: https://www.kaggle.com/datasets/elikplim/car-evaluation-data-set				
6A	Reinforcement Learning: Implement Reinforcement Learning using an example of a maze environment that the agent needs to explore.				

Assignment 1B:

Title of the Assignment: Feature Transformation:
Apply LDA Algorithm on Iris Dataset and classify which species a given flower belongs to.

Dataset Link: https://www.kaggle.com/datasets/uciml/iris

Dataset Description: The project involves using the Iris dataset, which includes measurements of sepal length, sepal width, petal length, and petal width for three species of iris flowers. The goal is to apply Linear Discriminant Analysis (LDA) to transform this data in a way that makes it easier to classify a given flower into one of the three species based on its measurements.

Objective of the Assignment: Students should be able to preprocess the dataset, apply the LDA algorithm for feature transformation, and build a classification model to classify iris flowers into different species.

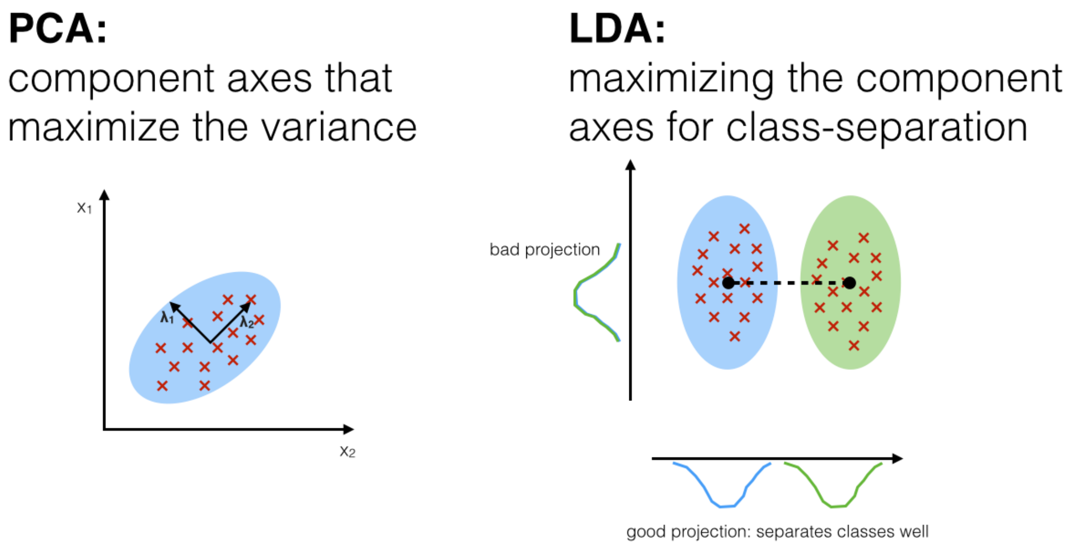
Theory:

Feature transformation is a critical step in preparing data for machine learning tasks. Linear Discriminant Analysis (LDA) is a dimensionality reduction technique that aims to find a lower-dimensional representation of the data while maximizing the separation between different classes. In the case of the Iris dataset, LDA can help us transform the feature space so that the species of iris flowers become more distinguishable.

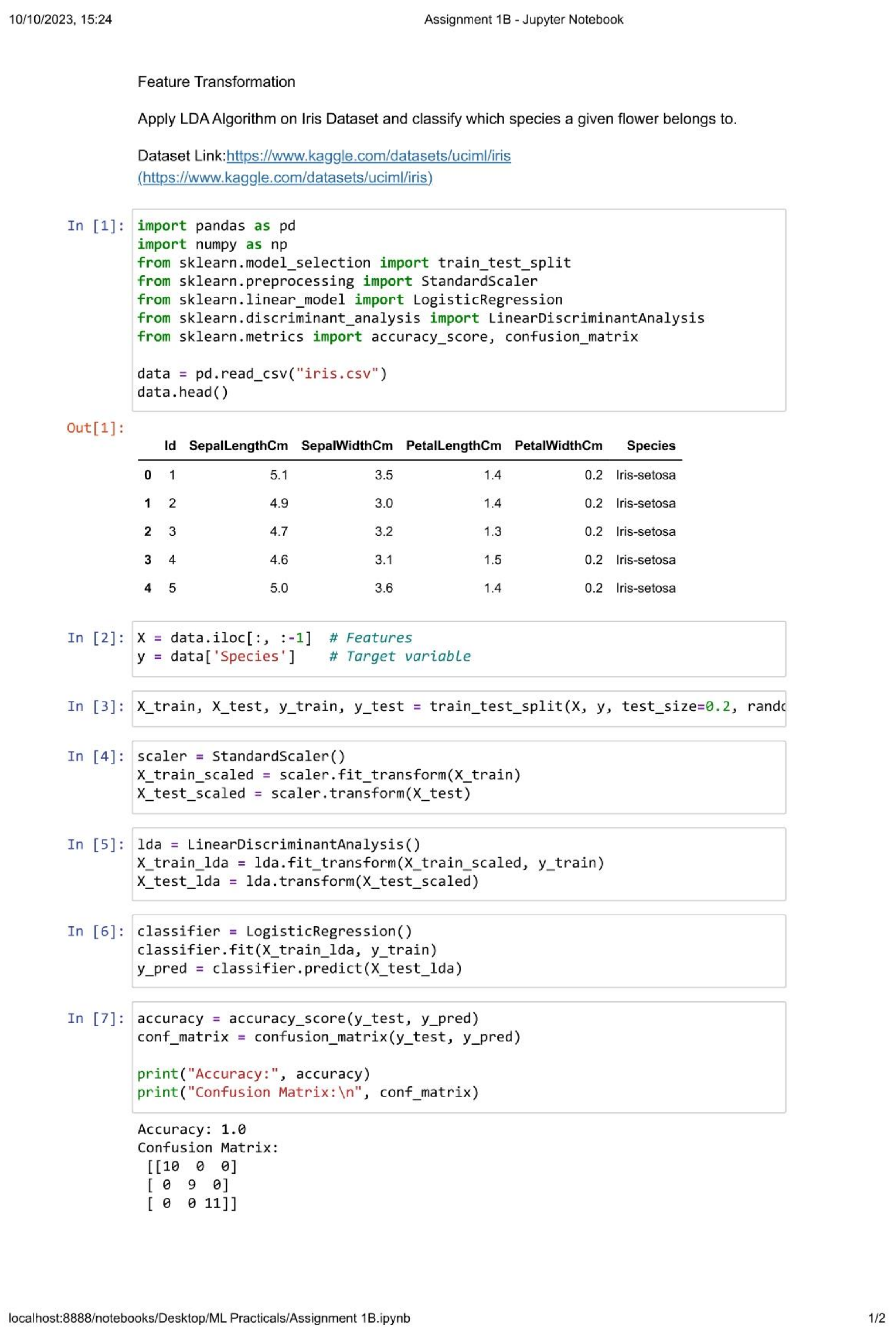
Linear Discriminant Analysis (LDA):
LDA is a supervised dimensionality reduction technique that seeks to find a linear combination of features that best separates multiple classes in the data. It aims to reduce the dimensionality of the data while preserving as much class discrimination information as possible.

Steps to Apply LDA in Python for Feature Transformation and Classification:

Step 1: Import necessary libraries
Step 2: Load the Iris dataset
Step 3: Preprocess the data (e.g., handle missing values, encode categorical variables if any)
Step 4: Apply Linear Discriminant Analysis (LDA) to transform the features
Step 5: Split the data into training and testing sets
Step 6: Train a classification model (e.g., logistic regression, decision tree, or support vector machine) on the transformed features
Step 7: Evaluate the model's performance using appropriate metrics (e.g., accuracy, precision, recall)
Step 8: Visualize the results (e.g., scatter plots of LDA-transformed features)


PCA:
component axes that maximize the variance
x_1
x_2
λ_1
λ_2
LDA:
maximizing the component axes for class-separation
bad projection
good projection: separates classes well

Conclusion: In this assignment, we have explored the concept of feature transformation using the LDA algorithm for dimensionality reduction and classification. By transforming the Iris dataset's features, we can build a classification model that can predict the species of iris flowers based on their measurements. This demonstrates the power of feature transformation techniques in improving the performance of machine learning models.



Feature Transformation

Apply LDA Algorithm on Iris Dataset and classify which species a given flower belongs to.

Dataset Link:https://www.kaggle.com/datasets/uciml/iris (https://www.kaggle.com/datasets/uciml/iris)

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import accuracy_score, confusion_matrix

data = pd.read_csv("iris.csv")
data.head()
```

Out[1]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [2]:
```
X = data.iloc[:, :-1]  # Features
y = data['Species']    # Target variable
```

In [3]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [4]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [5]:
```
lda = LinearDiscriminantAnalysis()
X_train_lda = lda.fit_transform(X_train_scaled, y_train)
X_test_lda = lda.transform(X_test_scaled)
```

In [6]:
```
classifier = LogisticRegression()
classifier.fit(X_train_lda, y_train)
y_pred = classifier.predict(X_test_lda)
```

In [7]:
```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
```

```
Accuracy: 1.0
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Assignment 2B:

Title of the Assignment: Regression Analysis
Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following:
a. Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness, and Kurtosis
b. Bivariate analysis: Linear and logistic regression modelling
c. Multiple Regression analysis
d. Also compare the results of the above analysis for the two data sets

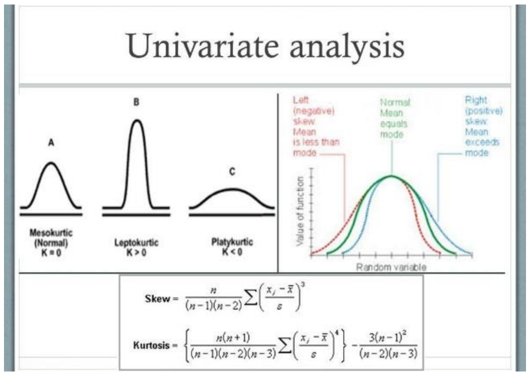
Dataset link: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

Dataset Description: The project involves working with two diabetes datasets: one from the UCI Machine Learning Repository and the other the Pima Indians Diabetes Database. The goal is to perform various regression analyses on these datasets, including univariate analysis to understand the statistical properties of the data, bivariate analysis involving linear and logistic regression modelling, and multiple regression analysis to predict outcomes. Finally, the results of these analyses will be compared between the two datasets.

Objective of the Assignment: Students should be able to conduct comprehensive regression analysis on real-world datasets, including univariate and bivariate analyses, as well as multiple regression modelling. Additionally, they should be able to compare the results of these analyses between two different datasets.

Theory:

Univariate Analysis:
Univariate analysis is the process of analyzing a single variable or attribute at a time. It involves computing summary statistics like frequency, mean, median, mode, variance, standard deviation, skewness, and kurtosis for a single variable. This analysis helps in understanding the distribution and characteristics of the data within that variable.



$$\text{Skew} = \frac{n}{(n-1)(n-2)}\sum\left(\frac{x_i-\bar{x}}{s}\right)^3$$

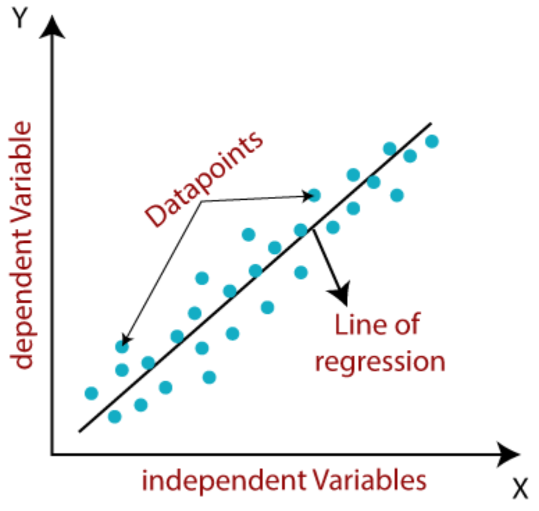
$$\text{Kurtosis} = \left\{\frac{n(n+1)}{(n-1)(n-2)(n-3)}\sum\left(\frac{x_i-\bar{x}}{s}\right)^4\right\} - \frac{3(n-1)^2}{(n-2)(n-3)}$$

Bivariate Analysis:

Bivariate analysis involves analyzing the relationship between two variables. In the context of regression analysis, it includes linear and logistic regression modelling. Linear regression is used when the dependent variable is continuous, while logistic regression is used when the dependent variable is categorical.

Multiple Regression Analysis:
Multiple regression analysis extends linear regression by considering multiple independent variables to predict the outcome. It helps in understanding how multiple predictors collectively influence the dependent variable.

Linear Regression:
Linear regression is a statistical method for modelling the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. It is commonly used for predicting numeric values.



Logistic Regression:
Logistic regression is a statistical method for modelling the probability of a binary outcome by fitting a logistic curve to the observed data. It is used for binary classification problems.

Conclusion: In this assignment, we have explored various aspects of regression analysis, including univariate and bivariate analysis, as well as multiple regression modelling.

Regression Analysis

Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following:

a. Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis b. Bivariate analysis: Linear and logistic regression modeling c. Multiple Regression analysis d. Also compare the results of the above analysis for the two data sets

Dataset link: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database (https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database)

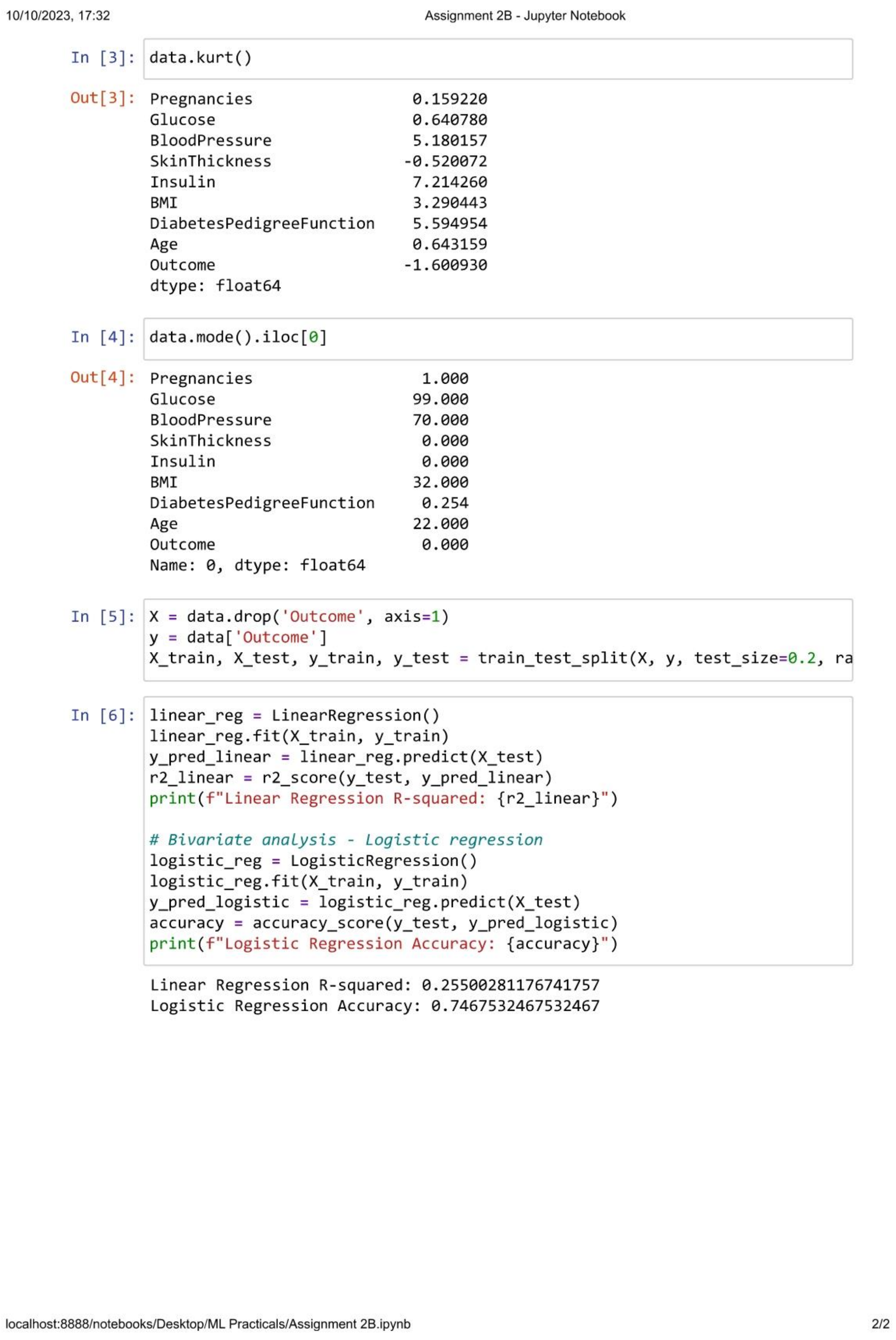
In [1]:
```
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import r2_score, accuracy_score
import warnings
warnings.filterwarnings("ignore")


# Load the diabetes dataset
data = pd.read_csv("diabetes.csv")
data.describe()
```

Out[1]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diat
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [2]:
```
data.skew()
```

Out[2]:
```
Pregnancies                 0.901674
Glucose                     0.173754
BloodPressure              -1.843608
SkinThickness               0.109372
Insulin                     2.272251
BMI                        -0.428982
DiabetesPedigreeFunction    1.919911
Age                         1.129597
Outcome                     0.635017
dtype: float64
```



In [3]:
```
data.kurt()
```

Out[3]:
```
Pregnancies                 0.159220
Glucose                     0.640780
BloodPressure               5.180157
SkinThickness              -0.520072
Insulin                     7.214260
BMI                         3.290443
DiabetesPedigreeFunction    5.594954
Age                         0.643159
Outcome                    -1.600930
dtype: float64
```

In [4]:
```
data.mode().iloc[0]
```

Out[4]:
```
Pregnancies                  1.000
Glucose                     99.000
BloodPressure               70.000
SkinThickness                0.000
Insulin                      0.000
BMI                         32.000
DiabetesPedigreeFunction     0.254
Age                         22.000
Outcome                      0.000
Name: 0, dtype: float64
```

In [5]:
```
X = data.drop('Outcome', axis=1)
y = data['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [6]:
```
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
y_pred_linear = linear_reg.predict(X_test)
r2_linear = r2_score(y_test, y_pred_linear)
print(f"Linear Regression R-squared: {r2_linear}")

# Bivariate analysis - Logistic regression
logistic_reg = LogisticRegression()
logistic_reg.fit(X_train, y_train)
y_pred_logistic = logistic_reg.predict(X_test)
accuracy = accuracy_score(y_test, y_pred_logistic)
print(f"Logistic Regression Accuracy: {accuracy}")
```

```
Linear Regression R-squared: 0.25500281176741757
Logistic Regression Accuracy: 0.7467532467532467
```

Assignment 3B:

Title of the Assignment: Classification Analysis:
Implement K-Nearest Neighbors (KNN) algorithm on social network ad dataset. Compute confusion matrix, accuracy, error rate, precision, and recall on the given dataset.

Dataset link: https://www.kaggle.com/datasets/rakeshrau/social-network-ads

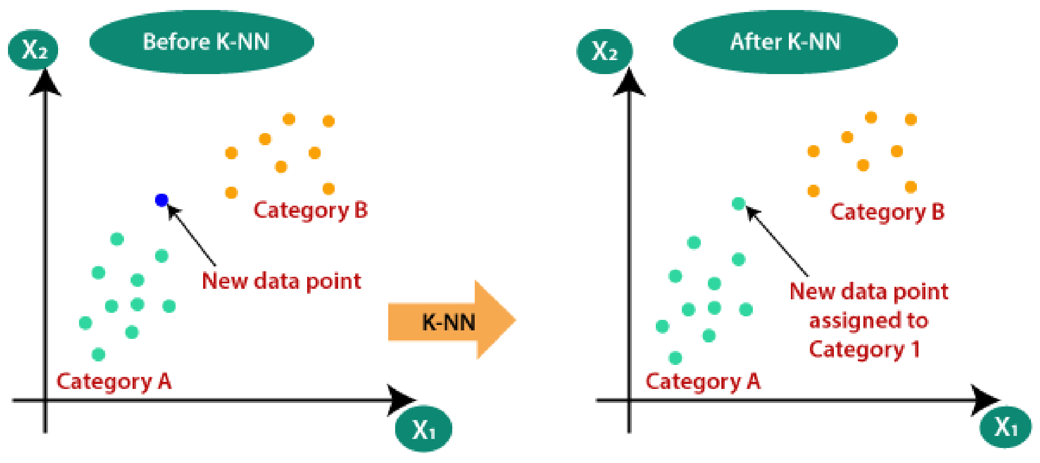
Dataset Description: In this project, we will work with a social network ad dataset that contains information about users, including features such as age, gender, and estimated salary, as well as whether a user clicked on a particular ad (0 for no, 1 for yes). The goal is to implement the K-Nearest Neighbors (KNN) algorithm on this dataset to predict whether a user is likely to click on the ad or not. We will compute various classification metrics, including confusion matrix, accuracy, error rate, precision, and recall.

Objective of the Assignment: Students should be able to apply the K-Nearest Neighbors algorithm to a real-world dataset for classification tasks. They should also learn how to evaluate the performance of a classification model using key metrics.

Theory:

K-Nearest Neighbors (KNN) Algorithm:

K-Nearest Neighbors is a supervised machine learning algorithm used for classification and regression tasks. In KNN, an object is classified by a majority vote of its neighbors, with the object being assigned to the class that is most common among its K nearest neighbors (K is a hyperparameter). KNN is a non-parametric and instance-based learning algorithm.



Confusion Matrix:

A confusion matrix is a table that is used to evaluate the performance of a classification model. It shows the true positive, true negative, false positive, and false negative values, which are essential for calculating various classification metrics.

Accuracy:

Accuracy measures the ratio of correctly predicted instances to the total instances in the dataset. It provides a general measure of the model's performance.

Error Rate:

Error rate is the complement of accuracy and measures the ratio of incorrectly predicted instances to the total instances. It provides the rate of misclassification.

Precision:

Precision is a metric that measures the accuracy of positive predictions. It is the ratio of true positive predictions to the total positive predictions and helps in assessing the model's ability to avoid false positives.

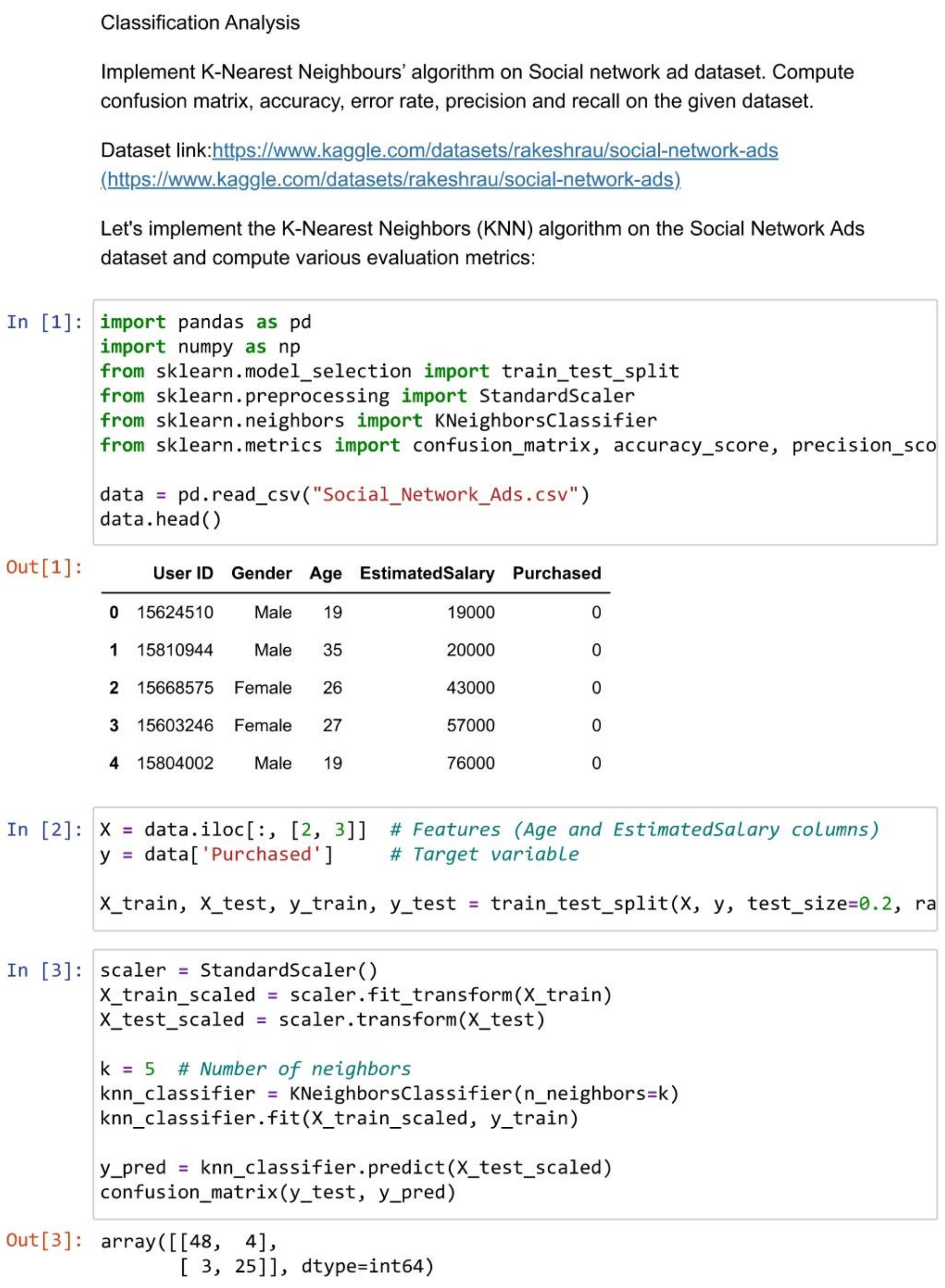
Recall:

Recall, also known as sensitivity or true positive rate, measures the ability of the model to correctly identify positive instances. It is the ratio of true positive predictions to the total actual positive instances.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Conclusion: In this assignment, we have explored the implementation of the K-Nearest Neighbors (KNN) algorithm on a social network ad dataset for classification. We have computed key classification metrics such as the confusion matrix, accuracy, error rate, precision, and recall to evaluate the model's performance in predicting user clicks on ads.



Classification Analysis

Implement K-Nearest Neighbours' algorithm on Social network ad dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.

Dataset link:https://www.kaggle.com/datasets/rakeshrau/social-network-ads (https://www.kaggle.com/datasets/rakeshrau/social-network-ads)

Let's implement the K-Nearest Neighbors (KNN) algorithm on the Social Network Ads dataset and compute various evaluation metrics:

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_sco

data = pd.read_csv("Social_Network_Ads.csv")
data.head()
```

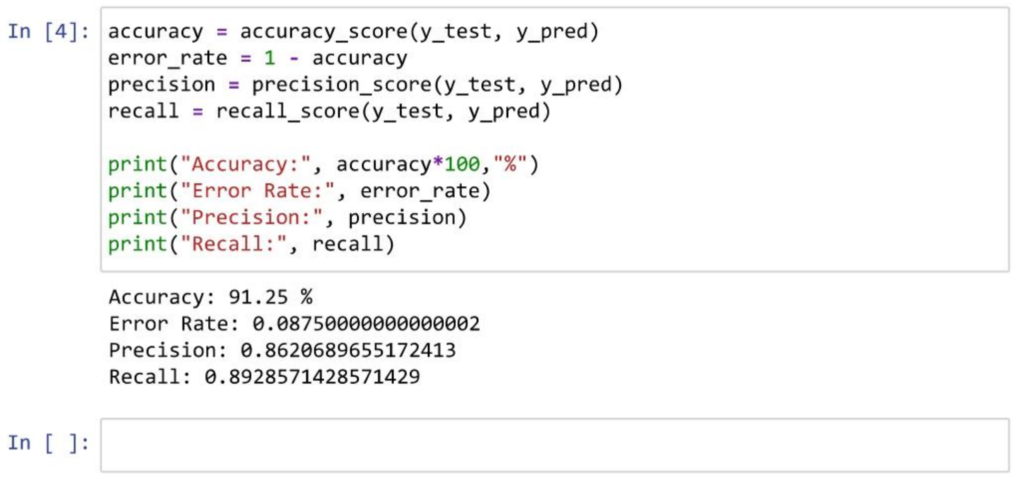
Out[1]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [2]:
```
X = data.iloc[:, [2, 3]]  # Features (Age and EstimatedSalary columns)
y = data['Purchased']     # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [3]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

k = 5  # Number of neighbors
knn_classifier = KNeighborsClassifier(n_neighbors=k)
knn_classifier.fit(X_train_scaled, y_train)

y_pred = knn_classifier.predict(X_test_scaled)
confusion_matrix(y_test, y_pred)
```

Out[3]:
```
array([[48,  4],
       [ 3, 25]], dtype=int64)
```



```
In [4]: accuracy = accuracy_score(y_test, y_pred)
        error_rate = 1 - accuracy
        precision = precision_score(y_test, y_pred)
        recall = recall_score(y_test, y_pred)

        print("Accuracy:", accuracy*100,"%")
        print("Error Rate:", error_rate)
        print("Precision:", precision)
        print("Recall:", recall)

        Accuracy: 91.25 %
        Error Rate: 0.08750000000000002
        Precision: 0.8620689655172413
        Recall: 0.8928571428571429

In [ ]:
```

Assignment 4A:

Title of the Assignment: Clustering Analysis:
Implement K-Means clustering on Iris.csv dataset. Determine the number of clusters using the elbow method.
Dataset Description: It has 150 entries with 1 dependent column and 4 feature columns.

Link for Dataset: https://www.kaggle.com/datasets/uciml/iris

Objective of the Assignment: Students should be able to implement K-means clustering on Iris.csv dataset. and determine the number of clusters using the Elbow method.

Theory:
K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of means clustering.

"It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties."

K Means Algorithm Working:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

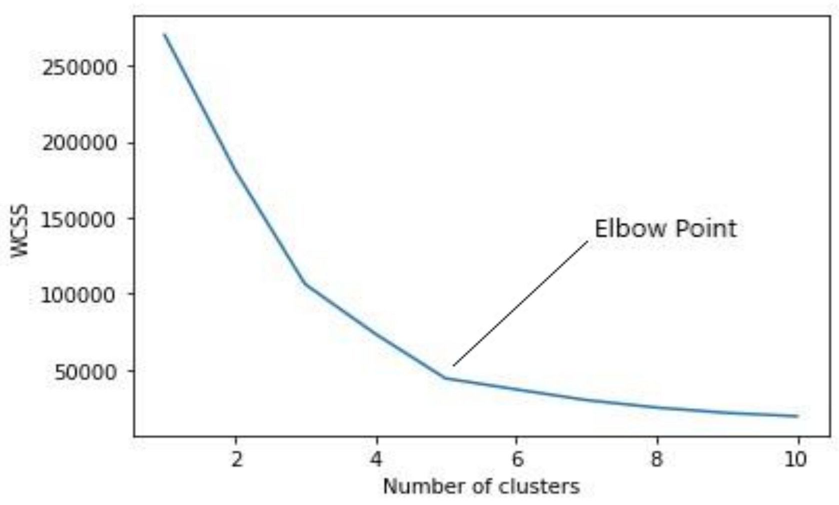
The elbow method is a popular unsupervised learning algorithm used in K-Means clustering. Unlike supervised learning, K-Means doesn't require labelled data. It involves randomly initializing K cluster centroids and iteratively adjusting them until they stop moving. Let's go through the steps involved in K-means clustering for a better understanding:

1. Select the number of clusters for the dataset (K)
2. Select the K number of centroids randomly from the dataset.
3. Now we will use Euclidean distance or Manhattan distance as the metric to calculate the distance of the points from the nearest centroid and assign the points to that nearest cluster centroid, thus creating K clusters.
4. Now we find the new centroid of the clusters thus formed.
5. Again, reassign the whole data point based on this new centroid, then repeat step 4. We will continue this for a given number of iterations until the position of the centroid doesn't change, i.e., there is no more convergence.

Finding the optimal number of clusters is an important part of this algorithm. A commonly used method for finding the optimum K value is **Elbow Method.**

K Means Clustering Using the Elbow Method

In the Elbow method, we are actually varying the number of clusters (K) from 1 – 10. For each value of K, we are calculating WCSS (Within-Cluster Sum of Square). WCSS is the sum of the squared distance between each point and the centroid in a cluster. When we plot the WCSS with the K value, the plot looks like an Elbow. As the number of clusters increases, the WCSS value will start to decrease. WCSS value is largest when K = 1. When we analyze the graph, we can see that the graph will rapidly change at a point and thus creating an elbow shape. From this point, the graph moves almost parallel to the X-axis. The K value corresponding to this point is the optimal value of K or an optimal number of clusters.



Conclusion:

In this way, we are studied the basic concepts of the K-Means Clustering algorithm in Machine Learning. We used the Elbow method to find the K-mean for clustering the data in our sample data set.

Clustering Analysis

Implement K-Means clustering on Iris.csv dataset. Determine the number of clusters using the elbow method.

Dataset Link: https://www.kaggle.com/datasets/uciml/iris (https://www.kaggle.com/datasets/uciml/iris)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

data = pd.read_csv("iris.csv")
data.head()
```

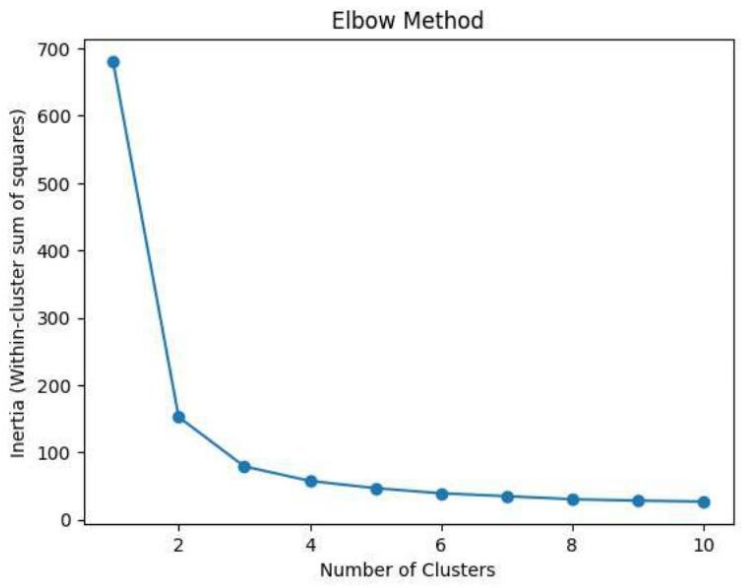
Out[1]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [2]:
```
X = data.iloc[:, [1, 2, 3, 4]]
```

In [3]:
```
inertia = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, max_iter=300, random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)
```

In [4]:
```
# Plot the Elbow Method graph
plt.plot(range(1, 11), inertia, marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia (Within-cluster sum of squares)')
plt.title('Elbow Method')
plt.show()
```



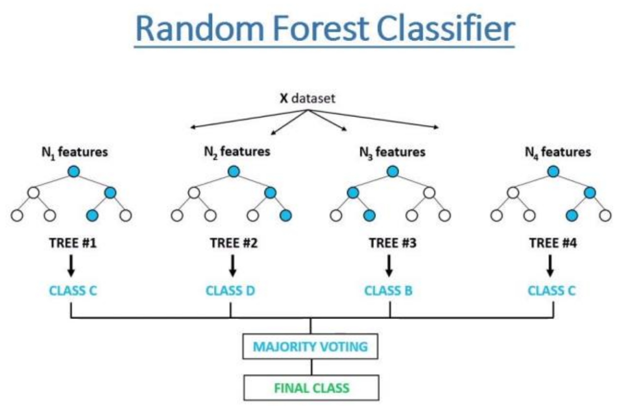
Assignment 5A:

Title of the Assignment: Ensemble Learning
Implement Random Forest Classifier model to predict the safety of the car.
Dataset link: https://www.kaggle.com/datasets/elikplim/car-evaluation-data-set

Dataset Description: In this project, we will work with a car evaluation dataset that contains information about various features of cars, such as their price, maintenance cost, number of doors, and safety rating. The goal is to implement the Random Forest Classifier model on this dataset to predict the safety level of the car. The safety level is categorized as "low," "medium," "high," and "very high."

Objective of the Assignment: Students should be able to apply ensemble learning techniques, specifically the Random Forest Classifier, to a real-world dataset for classification tasks. They should also learn how to evaluate the performance of an ensemble model.

Theory:
Random Forest Classifier:
Random Forest is an ensemble learning method used for classification and regression tasks. It is an ensemble of decision trees, where multiple decision trees are trained on different subsets of the data. The predictions from individual trees are combined to make the final prediction. Random Forest is known for its high accuracy and resistance to overfitting.



Ensemble Learning:
Ensemble learning is a machine learning technique that combines the predictions of multiple models to improve overall performance. Random Forest is an example of ensemble learning, where multiple decision trees are combined to make more robust predictions.

Evaluation Metrics:

Evaluation metrics are used to measure the performance of classification models. Common metrics include accuracy, precision, recall, and F1-score, among others. These metrics help assess the model's ability to make correct predictions and handle different aspects of classification performance.

Conclusion: In this assignment, we have explored the implementation of the Random Forest Classifier model on a car evaluation dataset to predict the safety level of cars. We have also discussed key ensemble learning concepts and evaluation metrics used to assess the model's performance.

Ensemble Learning

Implement Random Forest Classifier model to predict the safety of the car.

Dataset link: https://www.kaggle.com/datasets/elikplim/car-evaluation-data-set (https://www.kaggle.com/datasets/elikplim/car-evaluation-data-set)

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import category_encoders as ce
from sklearn.metrics import accuracy_score, confusion_matrix

data = pd.read_csv("car_evaluation.csv")
data.head()
```

Out[1]:

	vhigh	vhigh.1	2	2.1	small	low	unacc
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

In [2]:
```
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
data.columns = col_names

data.head()
```

Out[2]:

	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

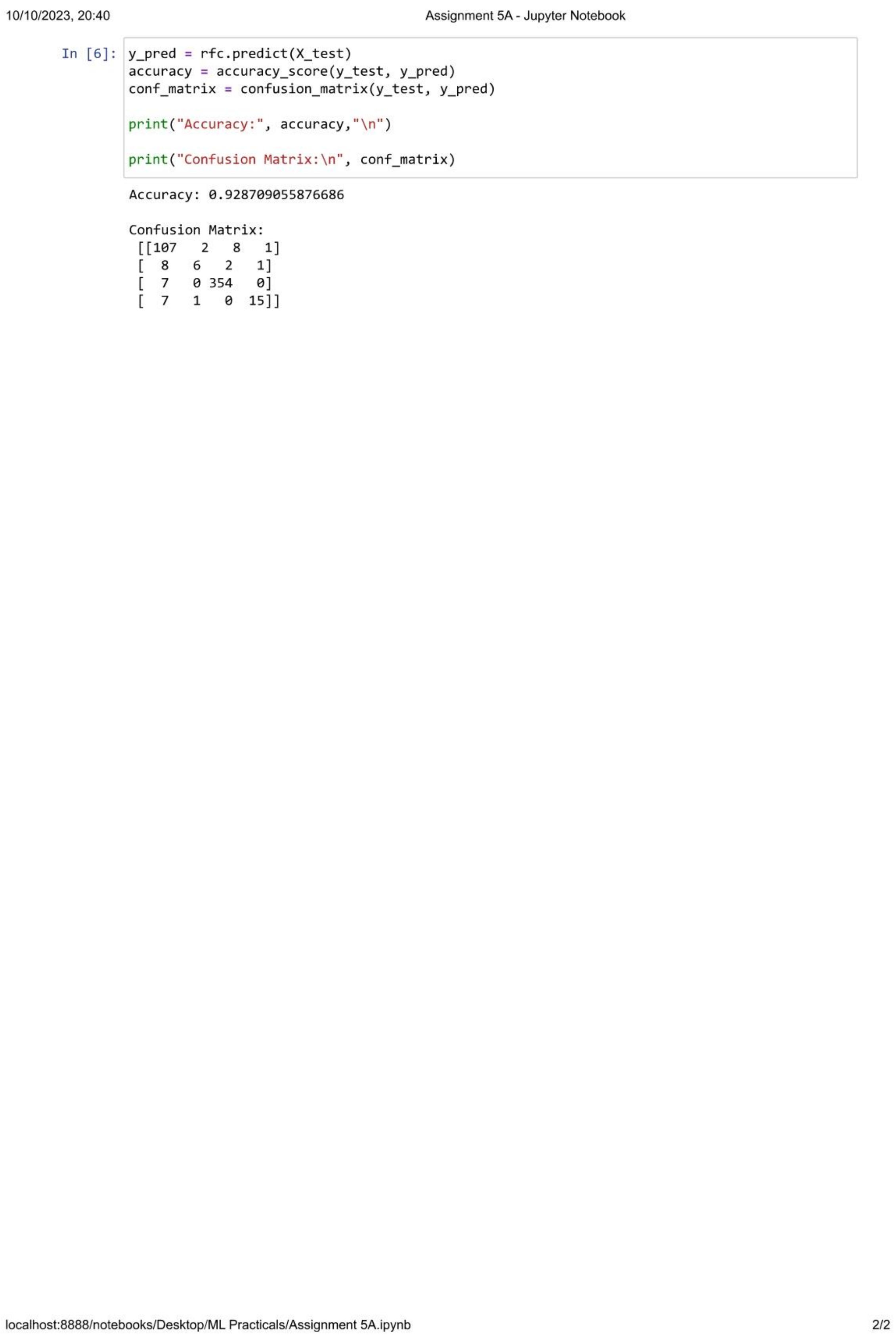
In [3]:
```
X =data.drop(['class'],axis=1)
y = data['class']

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42)
X_train.shape,X_test.shape
```

Out[3]:
```
((1208, 6), (519, 6))
```

In [4]:
```
encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safet
X_train = encoder.fit_transform(X_train)
X_test = encoder.transform(X_test)
```

In [5]:
```
rfc=RandomForestClassifier(random_state=0)
rfc.fit(X_train,y_train)
```

Out[5]:
```
▾        RandomForestClassifier
RandomForestClassifier(random_state=0)
```



10/10/2023, 20:40 Assignment 5A - Jupyter Notebook

```
In [6]: y_pred = rfc.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        conf_matrix = confusion_matrix(y_test, y_pred)

        print("Accuracy:", accuracy,"\n")

        print("Confusion Matrix:\n", conf_matrix)
```

```
Accuracy: 0.928709055876686

Confusion Matrix:
 [[107   2   8   1]
 [  8   6   2   1]
 [  7   0 354   0]
 [  7   1   0  15]]
```

localhost:8888/notebooks/Desktop/ML Practicals/Assignment 5A.ipynb 2/2

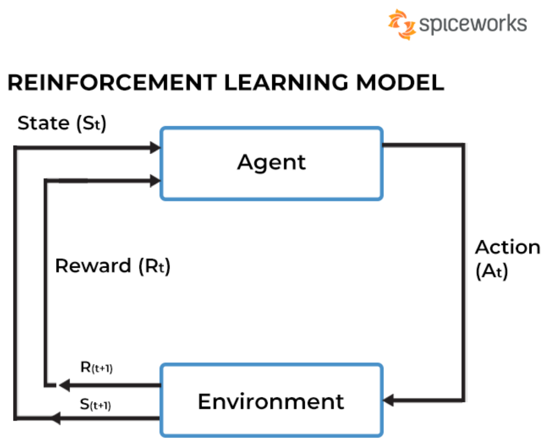
Assignment 6A:

Title of the Assignment: Reinforcement Learning
Implement Reinforcement Learning using an example of a maze environment that the agent needs to explore.

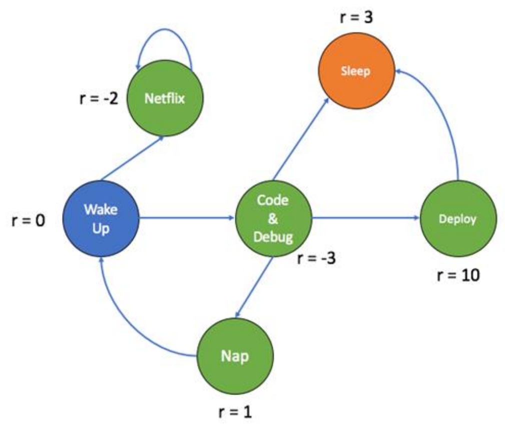
Assignment Description: In this project, we will delve into the field of Reinforcement Learning (RL) by implementing RL techniques using a maze environment. The primary objective is to create an RL agent capable of navigating and solving a maze problem autonomously. The agent will learn to make decisions and take actions based on its interactions with the maze environment, gradually improving its ability to navigate and reach a predefined goal.

Objective of the Assignment: The assignment aims to introduce students to the fundamentals of Reinforcement Learning, specifically in the context of solving maze problems. Students will learn how RL agents can learn and make decisions through interaction with their environment.

Theory:
Reinforcement Learning (RL):
Reinforcement Learning is a type of machine learning where an agent interacts with an environment and learns to make sequences of decisions to maximize a cumulative reward. It involves the concept of an agent learning from its experiences, taking actions to achieve a goal, and receiving feedback in the form of rewards or penalties.



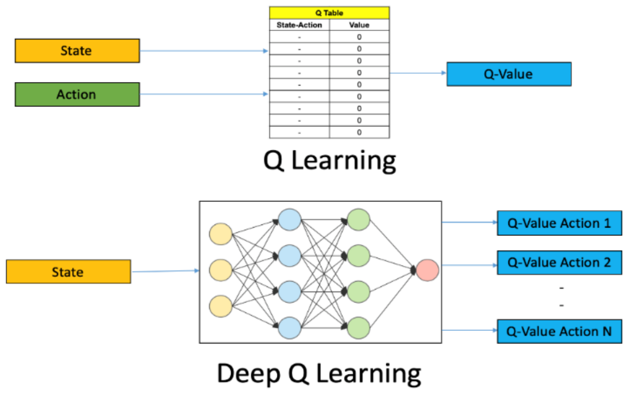
Markov Decision Process (MDP):
MDP is a mathematical framework used to describe the RL problem. It defines the agent's interactions with the environment as a sequence of states, actions, and rewards, where the transition from one state to another depends on the agent's actions.



Agent, Environment, and State:
In RL, there are typically three main components: the agent (learner), the environment (with which the agent interacts), and the state (representing the current situation of the environment). The agent selects actions, the environment responds, and the state changes accordingly.

Actions, Rewards, and Policies:
Actions represent the decisions made by the agent, rewards are the feedback given by the environment to guide the agent, and policies are strategies or rules that the agent follows to select actions.

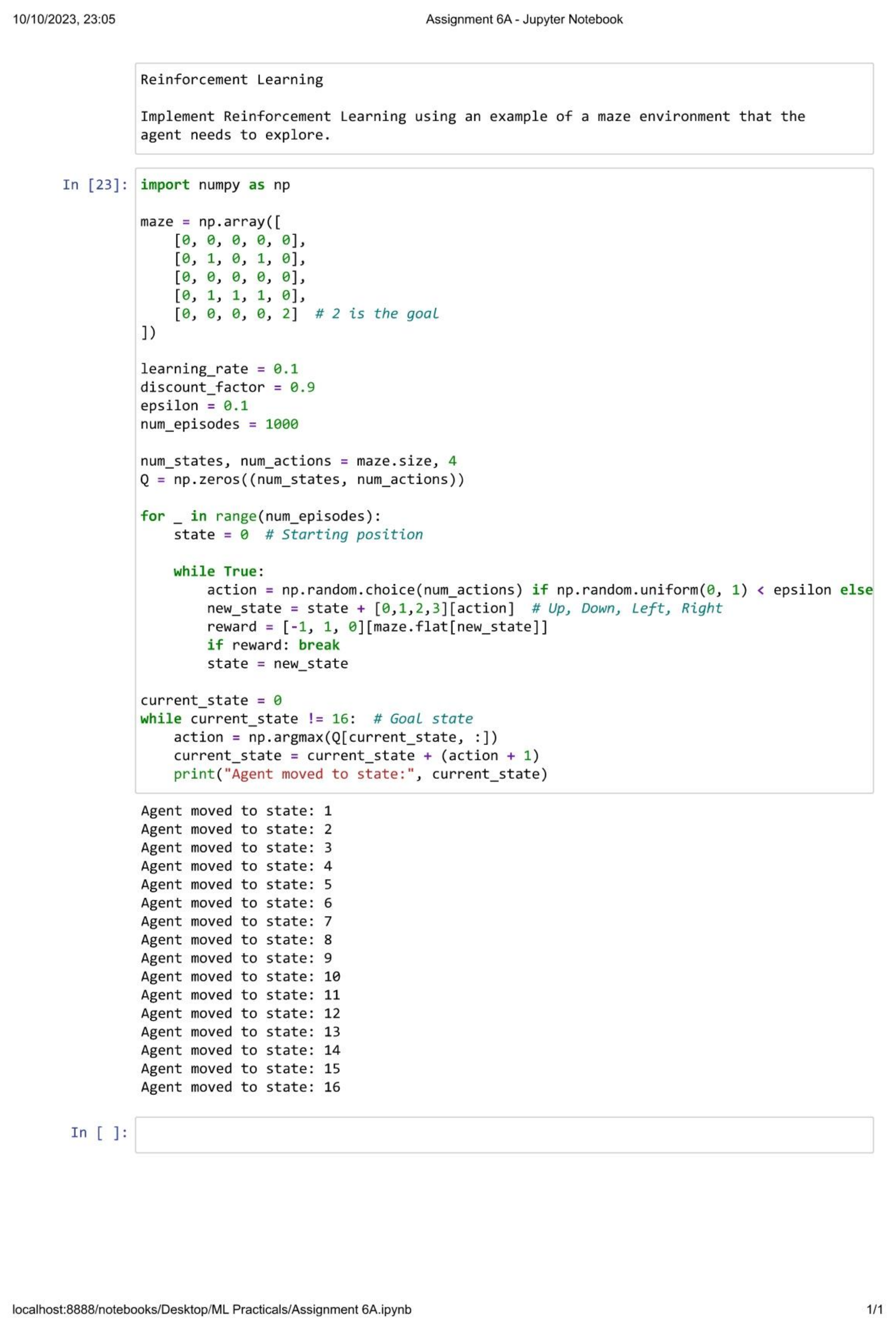
Q-Learning and Value Iteration:
Q-Learning is a popular RL algorithm used to find optimal policies for decision-making in an MDP. Value Iteration is another technique used to compute the expected cumulative rewards for each state-action pair.



Exploration vs. Exploitation:
One of the key challenges in RL is the trade-off between exploration (trying new actions to learn) and exploitation (choosing known actions for immediate reward). Balancing these aspects is crucial for successful RL.

Learning from Experience (Episodes):
In RL, an agent typically learns from a series of episodes, where each episode consists of a sequence of actions, states, and rewards. The agent uses these experiences to update its policy and improve its decision-making.

Conclusion: This assignment will provide students with a practical understanding of Reinforcement Learning by implementing an RL agent to navigate and solve a maze environment.



10/10/2023, 23:05 Assignment 6A - Jupyter Notebook

```
Reinforcement Learning

Implement Reinforcement Learning using an example of a maze environment that the
agent needs to explore.
```

In [23]:
```
import numpy as np

maze = np.array([
    [0, 0, 0, 0, 0],
    [0, 1, 0, 1, 0],
    [0, 0, 0, 0, 0],
    [0, 1, 1, 1, 0],
    [0, 0, 0, 0, 2]  # 2 is the goal
])

learning_rate = 0.1
discount_factor = 0.9
epsilon = 0.1
num_episodes = 1000

num_states, num_actions = maze.size, 4
Q = np.zeros((num_states, num_actions))

for _ in range(num_episodes):
    state = 0  # Starting position

    while True:
        action = np.random.choice(num_actions) if np.random.uniform(0, 1) < epsilon else
        new_state = state + [0,1,2,3][action]  # Up, Down, Left, Right
        reward = [-1, 1, 0][maze.flat[new_state]]
        if reward: break
        state = new_state

current_state = 0
while current_state != 16:  # Goal state
    action = np.argmax(Q[current_state, :])
    current_state = current_state + (action + 1)
    print("Agent moved to state:", current_state)
```

```
Agent moved to state: 1
Agent moved to state: 2
Agent moved to state: 3
Agent moved to state: 4
Agent moved to state: 5
Agent moved to state: 6
Agent moved to state: 7
Agent moved to state: 8
Agent moved to state: 9
Agent moved to state: 10
Agent moved to state: 11
Agent moved to state: 12
Agent moved to state: 13
Agent moved to state: 14
Agent moved to state: 15
Agent moved to state: 16
```

In [ ]:

localhost:8888/notebooks/Desktop/ML Practicals/Assignment 6A.ipynb 1/1