|  |  |  |  |
| --- | --- | --- | --- |
| A picture containing drawing, stop, room  Description automatically generated | Business Intelligence  Practical #7 | | |
|  |  |  |  |
| **Name** | Farman Kazi | **Roll Number** | 21302B0049 |
| **Class** | TY B. Sc. IT | **Division** | A |
| **Subject/Course:** | Business intelligence | | |
| **Topic** | Decision Tree Classifier | | |
|  |  |  |  |
| **Overview of Decision Tree Classifier**  **What are the steps to perform Decision Tree Classifier?** | | | |
| 1. **Import Libraries:**    1. **pandas**: for data manipulation and handling dataframes.    2. **tree** and **DecisionTreeClassifier** from **sklearn**: for building and working with decision trees.    3. **pydotplus**: for visualizing the decision tree.    4. **matplotlib.pyplot** and **matplotlib.image**: for plotting and displaying the decision tree image. 2. **Load Data:**    1. Reads the data from a CSV file named "shows.csv" using **pandas.read\_csv()**.    2. Assumes the CSV file contains information relevant to making a decision about someone going to a show (e.g., Age, Experience, Rank, Nationality, and a Go/No Go decision). 3. **Data Preprocessing:**    1. Creates a dictionary mapping string values in the "Nationality" column to numerical values (0, 1, 2).    2. Similarly, maps "YES" and "NO" values in the "Go" column to 1 and 0 respectively.    3. This step is crucial for decision tree algorithms as they work better with numerical features. 4. **Separate Features and Target:**    1. Defines a list named "features" containing the column names of the features used for prediction (Age, Experience, Rank, Nationality).    2. Separates the features (columns specified in "features") into a dataframe named "**X**".    3. Extracts the target variable ("Go") into a separate dataframe named "**Y**". 5. **Train the Decision Tree Model:**    1. Creates a **DecisionTreeClassifier** object from **sklearn.tree**.    2. Fits the model to the training data using **dtree.fit(X, Y)**. This process builds the decision tree based on the features and target variable. 6. **Visualize the Decision Tree:**    1. Uses **tree.export\_graphviz()** to export the decision tree structure in a DOT language format suitable for visualization.    2. Creates a graph object from the DOT data using **pydotplus.graph\_from\_dot\_data**.    3. Saves the graph as a PNG image named "mydecisiontree.png" using **graph.write\_png**.    4. Loads the image using **matplotlib.image.imread**.    5. Displays the image using **plt.imshow** and **plt.show**. 7. **Make Predictions:**    1. Creates a sample data point with features (Age, Experience, Rank, Nationality) to predict the decision (Go/No Go).    2. Uses **dtree.predict** to predict the target variable (Go) for the new data point.    3. Repeats the prediction for another data point with slightly different features. | | | |
| import pandas from sklearn import tree import pydotplus from sklearn.tree import DecisionTreeClassifier import matplotlib.pyplot as plt import matplotlib.image as pltimg  df = pandas.read\_csv("/shows.csv") print(df)    d = {'UK': 0, 'USA': 1, 'N': 2} df['Nationality'] = df['Nationality'].map(d) d = {'YES': 1, 'NO': 0} df['Go'] = df['Go'].map(d) print(df)    features = ['Age', 'Experience', 'Rank', 'Nationality'] X = df[features] Y = df['Go'] print(X) print(Y)    dtree = DecisionTreeClassifier() dtree = dtree.fit(X, Y) data = tree.export\_graphviz(dtree, out\_file = None, feature\_names = features) graph = pydotplus.graph\_from\_dot\_data(data) graph.write\_png('mydecisiontree.png') img = pltimg.imread('mydecisiontree.png') imgplot = plt.imshow(img) plt.show()    print(dtree.predict([[40, 10, 7, 1]]))    print(dtree.predict([[40, 10, 6, 1]])) | | | |