decision_tree_classification

April 1, 2024

1 Decision Tree Classification

1.1 Importing the libraries

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 1
----> 1 import numpy as np
      2 import matplotlib.pyplot as plt
      3 import pandas as pd

ModuleNotFoundError: No module named 'numpy'
```

1.2 Importing the dataset

[ ]:
```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

1.3 Splitting the dataset into the Training set and Test set

[ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,␣
↪random_state = 0)
```

[ ]:
```
print(X_train)
```

[ ]:
```
print(y_train)
```

[ ]:
```
print(X_test)
```

[ ]:
```
print(y_test)
```

1.4 Feature Scaling

```
[ ]: from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

```
[ ]: print(X_train)
```

```
[ ]: print(X_test)
```

1.5 Training the Decision Tree Classification model on the Training set

```
[ ]: from sklearn.tree import DecisionTreeClassifier
     classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
     classifier.fit(X_train, y_train)
```

1.6 Predicting a new result

```
[ ]: print(classifier.predict(sc.transform([[30,87000]])))
```

1.7 Predicting the Test set results

```
[ ]: y_pred = classifier.predict(X_test)
     print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.
       ↪reshape(len(y_test),1)),1))
```

1.8 Making the Confusion Matrix

```
[ ]: from sklearn.metrics import confusion_matrix, accuracy_score
     cm = confusion_matrix(y_test, y_pred)
     print(cm)
     accuracy_score(y_test, y_pred)
```

```
[ ]: from matplotlib import pyplot as plt
     from sklearn import datasets
     from sklearn.tree import DecisionTreeClassifier
     from sklearn import tree


     fig = plt.figure(figsize=(25,20))
     _ = tree.plot_tree(classifier,
                        feature_names = ['Age' , 'EstimatedSalary'],
                        class_names = ['Yes', 'No'],
                        filled=True)
```

Q1. Run the above steps and make sure that you understand them.

Q2. Use the decision tree classification on the diabetes data set and compare the accuracy with Logistic Regression and kNN.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, accuracy_score

# Load the dataset
dataset = pd.read_csv('diabetes.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=0)

# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Train the Decision Tree classifier
classifier = DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(X_train, y_train)

# Predict the test set results
y_pred = classifier.predict(X_test)

# Evaluate the classifier
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Plot the decision tree
plt.figure(figsize=(20, 15))
plot_tree(classifier,
          ␣
 ↪feature_names=['pregnant','glucose','bp','skin','insulin','bmi','pedigree','age'],
          class_names=['No', 'Yes'],
          filled=True)
plt.show()
```

Q3. Download data from https://www.kaggle.com/aljarah/xAPI-Edu-Data

Q4. Study the data to understand the independent and dependent variables

Q5. Use 60% of the data for training and 40% of the data for testing

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix

# Load the dataset
df = pd.read_csv("xAPI-Edu-Data.csv")

# Split features and target variable
X = df.drop('Class', axis=1)
y = df['Class']

# One-hot encode categorical features
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), ['gender',␣
 ↪'NationalITy', 'PlaceofBirth', 'StageID', 'GradeID', 'SectionID', 'Topic',␣
 ↪'Semester', 'Relation', 'ParentAnsweringSurvey', 'ParentschoolSatisfaction',␣
 ↪'StudentAbsenceDays'])], remainder='passthrough')
X = ct.fit_transform(X)

# Convert sparse matrix to dense array
X = X.toarray()

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,␣
 ↪random_state=0)
```

Q6. Create a decision tree to decide on whether the students are classified into low / middle / high level

[ ]:
```
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Q7. Test your decision tree with the test data

[ ]:
```
y_pred = classifier.predict(X_test)
```

Q8. Plot the tree

[ ]:
```
plt.figure(figsize=(20, 15))
tree.plot_tree(classifier, filled=True)
plt.show()
```

Q9. Create a confusion matrix

[ ]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

[1]:
```

```