```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Load the dataset
dataset = pd.read_csv("/home/avcoe/Admission_Predict.csv")

# Display basic information about the dataset
print(dataset.info())
print(f"Size of dataset: {dataset.size}")
print(f"Shape of dataset: {dataset.shape}")
print(dataset.describe())

# Display data type of each column
print("Data Type for Each Columns are\n", dataset.dtypes.value_counts())

# Display missing values
print(dataset.isnull().sum().sort_values(ascending=False)/len(dataset))

# Display correlation matrix
corr_matrix = dataset.corr()
sns.heatmap(corr_matrix, annot=True, linewidths=0.5, fmt=".2f", cmap="YlGnBu")

# Identify categorical and continuous features
categorical_val = [col for col in dataset.columns if dataset[col].nunique() <= 10]
```

```
categorical_val.remove('target')

# One-hot encode categorical features
dataset = pd.get_dummies(dataset, columns=categorical_val)

# Standardize selected columns using StandardScaler
s_sc = StandardScaler()
col_to_scale = ['GRE Score', 'CGPA']
dataset[col_to_scale] = s_sc.fit_transform(dataset[col_to_scale])

# Split the data into training and testing sets
X = dataset.drop('target', axis=1)
y = dataset.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Classification using Decision Tree Classifier
tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)

# Print classification results
def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    data = (X_train, y_train) if train else (X_test, y_test)
    pred = clf.predict(data[0])
    clf_report = pd.DataFrame(classification_report(data[1], pred, output_dict=True))
    data_type = "Train" if train else "Test"

    print(f"{data_type} Result:\n====================================================")
    print(f"Accuracy Score: {accuracy_score(data[1], pred) * 100:.2f}%")
    print("_______________________________________________________")
    print(f"CLASSIFICATION REPORT:\n{clf_report}")
    print("____________________________________________________")
    print(f"Confusion Matrix:\n{confusion_matrix(data[1], pred)}\n")

print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)
```