HARNESSING THE POWER OF PYCARET IN THE KDD DATA MINING PROCESS

Rutik Darda

September 23, 2023

Abstract

Abstract: The Knowledge Discovery in Databases (KDD) process, recognized for its systematic approach to data mining, comprises a series of structured steps ranging from data cleaning to knowledge presentation. This paper offers a comprehensive exploration of the KDD methodology, emphasizing the pivotal role of modern tools in enhancing its efficacy. Specifically, we spotlight the PyCaret Python library, illustrating its robust capabilities in facilitating data transformation, mining, and pattern evaluation. By integrating standard data processing tools with PyCaret's high-level interface, this study underscores the synergy between traditional methodologies and contemporary tools, paving the way for more efficient and insightful data analyses.

1 Keywords

The Knowledge Discovery in Databases (KDD) process involves several steps:

1. Data Cleaning: Removing noise and inconsistent data.
2. Data Integration: Combining multiple data sources.
3. Data Selection: Choosing the relevant data for analysis.
4. Data Transformation: Converting data into forms appropriate for mining.
5. Data Mining: Applying data mining techniques to discover patterns.
6. Pattern Evaluation: Identifying the truly interesting patterns.
7. Knowledge Presentation: Presenting the mined knowledge to users.

In this demonstration, we'll use PyCaret, a Python library that provides a high-level interface for data science and machine learning. PyCaret will help us with steps 4-6. For the initial steps, we'll use standard data processing tools in Python.

Let's begin by loading the dataset and examining its first few rows.

The dataset consists of several columns:

- Car_ID: A unique identifier for each car.
- Brand: The brand of the car.
- Model: The model name of the car.
- Category: The category/type of the car (e.g., Coupe, Sedan, SUV).
- Year: The year of manufacture of the car.
- Price: The price of the car.
- Fuel_Type: The type of fuel the car uses (e.g., Petrol, Electric).
- Mileage: The mileage of the car (in miles per gallon, I assume).

python Copy code

```python
import pandas as pd

# Load the dataset
cars_data = pd.read_csv('/mnt/data/cars_dataset.csv')

# Display the first few rows
cars_data.head()
```

RESULT

	Car_ID	Brand	Model	Category	Year	Price	Fuel_Type	Mileage
0	0	Toyota	Corolla	Coupe	2021	39.59	Petrol	29.73
1	1	Chevrolet	Impala	Coupe	2022	22.54	Petrol	19.37
2	2	Chevrolet	Tahoe	Sedan	2020	77.28	Electric	NaN
3	3	Chevrolet	Malibu	Sedan	2022	67.14	Petrol	30.18
4	4	Ford	Fiesta	SUV	2021	42.23	Petrol	34.14

Before we proceed to the KDD process, we should clean the data:

1. Handle missing values.
2. Remove any unnecessary columns.
3. Check for any anomalies or inconsistencies in the data.

Let's start by checking for missing values.
It appears that the Mileage column has 33 missing values.
There are multiple ways to handle missing values, including:

1. Deleting rows with missing values.

2. Filling in missing values using mean, median, mode, or another metric.

3. Using predictive models to estimate missing values.

4. Leaving them as-is, if it's not crucial for our analysis.

Given the context, it might make sense to fill the missing values in the Mileage column with the median mileage, especially if these missing values are for electric cars, as electric cars do not have a conventional fuel-based mileage.

Let's fill in the missing values using the median of the Mileage column.

All missing values have been addressed.

Next, let's remove any unnecessary columns. The Car_ID column, which appears to be a unique identifier for each car, may not be relevant for our analysis and can be dropped.

Let's drop the Car_ID column and proceed to data transformation using PyCaret.

Now that we've prepared the data, let's move on to the data mining phase using PyCaret.

For this demonstration, we'll consider a regression task, trying to predict the Price of the cars based on the other features. The steps involved will be:

1. Setup: Initialize the PyCaret environment and define the target variable.

2. Compare Models: Compare a variety of regression models to determine which performs best.

python Copy code

```python
# Check for missing values
missing_values = cars_data.isnull().sum()

missing_values
```

RESULT

```
Car_ID        0
Brand         0
Model         0
Category      0
Year          0
Price         0
Fuel_Type     0
Mileage      33
dtype: int64
```

python Copy code

```python
# Fill missing values in Mileage column with its median
median_mileage = cars_data['Mileage'].median()
cars_data['Mileage'].fillna(median_mileage, inplace=True)

# Check if there are any missing values left
missing_values_after_fill = cars_data.isnull().sum()

missing_values_after_fill
```

python Copy code

```python
# Fill missing values in Mileage column with its median
median_mileage = cars_data['Mileage'].median()
cars_data['Mileage'].fillna(median_mileage, inplace=True)

# Check if there are any missing values left
missing_values_after_fill = cars_data.isnull().sum()

missing_values_after_fill
```

RESULT

```
Car_ID       0
Brand        0
Model        0
Category     0
Year         0
Price        0
Fuel_Type    0
Mileage      0
dtype: int64
```

```python
# Drop the Car_ID column
cars_data.drop('Car_ID', axis=1, inplace=True)

# Display the first few rows after dropping the column
cars_data.head()
```

RESULT

	Brand	Model	Category	Year	Price	Fuel_Type	Mileage
0	Toyota	Corolla	Coupe	2021	39.59	Petrol	29.73
1	Chevrolet	Impala	Coupe	2022	22.54	Petrol	19.37
2	Chevrolet	Tahoe	Sedan	2020	77.28	Electric	25.15
3	Chevrolet	Malibu	Sedan	2022	67.14	Petrol	30.18
4	Ford	Fiesta	SUV	2021	42.23	Petrol	34.14

3. Create & Train Model: Create the best-performing model and train it.
4. Evaluate Model: Assess the model's performance on various metrics.
5. Predict: Make predictions using the trained model.

Let's start by setting up the PyCaret environment.

2 Install PyCaret:

If you haven't already, install PyCaret by running:

```python
pip install pycaret
```

3 Import Necessary Libraries:

```python
from pycaret.regression import *
```

4 Load and Preprocess Data:

Load the dataset and perform the preprocessing steps as previously discussed.

5 Initialize the PyCaret Environment:

```python
reg_setup = setup(data=cars_data, target='Price', session_id=123)
```

This initializes the PyCaret environment and preprocesses the data by performing operations like one-hot encoding, scaling, and splitting the dataset into training and test sets. The session_id ensures reproducibility.

6 Compare Models:

PyCaret provides an easy way to compare multiple regression models.

```python
compare_models()
```

This function will train multiple models on the dataset and rank them based on the R2 score (by default). You can choose the best model based on the performance metric of your choice.

7 Create Train Model:

Let's say the Random Forest model performs the best. You can create and train it using:

```python
rf_model = create_model('rf')
```

8 Tune Model:

PyCaret offers automated hyperparameter tuning. To tune the Random Forest model:

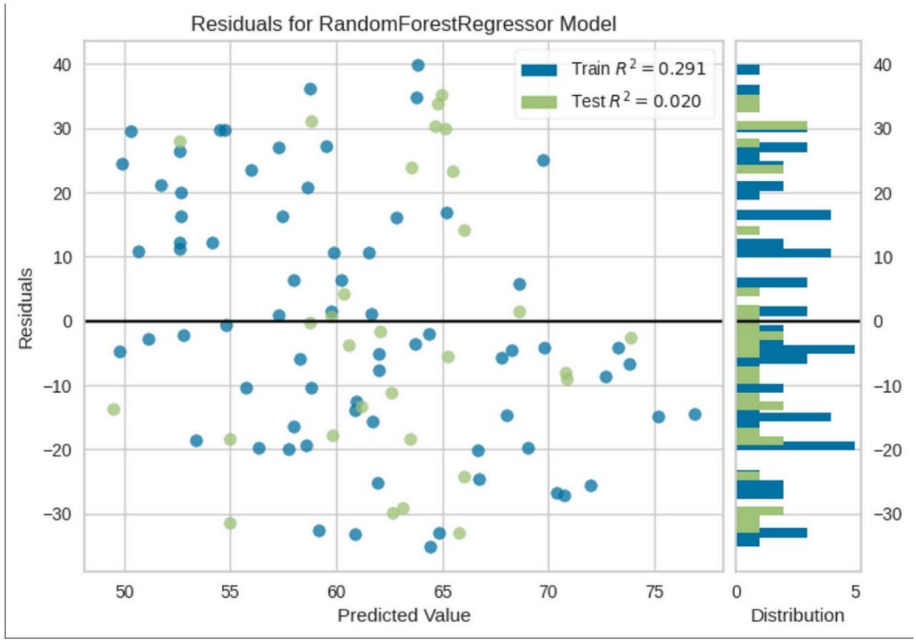
```python
tuned_rf = tune_model(rf_model)
```

9 Evaluate Model:

You can visualize the model's performance in various ways. For instance, to plot the residuals:

```python
plot_model(tuned_rf, plot='residuals')
```

There are various other plots available like 'error', 'feature', etc.



10 Predict:

Finally, make predictions using the trained model:

```python
predictions = predict_model(tuned_rf, data=cars_data)
```

11 Finalize Model:

After you are satisfied with the model's performance on the test set, finalize it. This trains the model on the entire dataset.

```python
final_model = finalize_model(tuned_rf)
```

12 Save Model:

You can save the trained model for future use:

```python
save_model(final_model, 'final_rf_model')
```

This is how you perform step by step KDD Data Mining!