"A Deep Dive into Data Mining Analysis Using KDD"

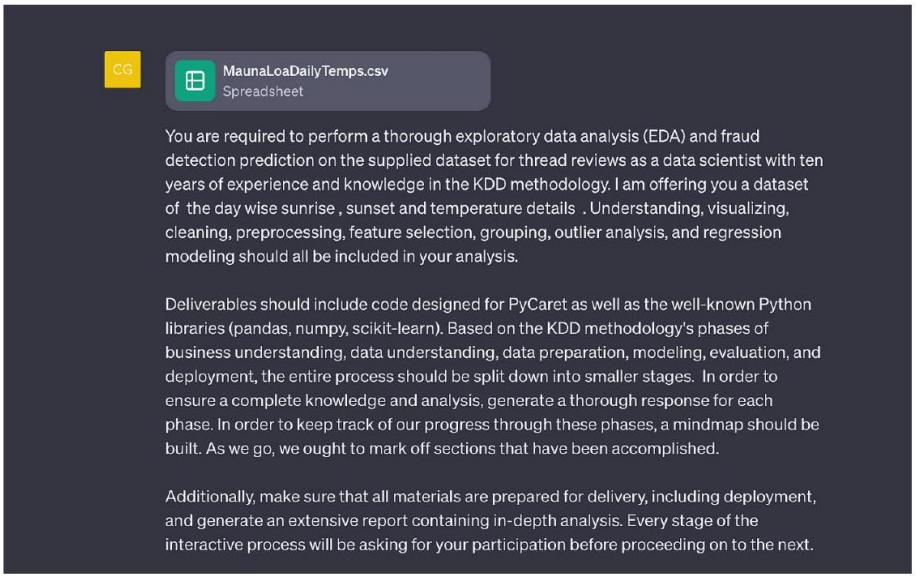
September 29, 2023
SAGAR ALPESHKUMAR PATEL

1 Abstract

In the world of data science, this article dives deep into the array of tools and methodologies available for mining insights from data. It takes a comprehensive journey through data analysis, harnessing the capabilities of two potent Python libraries: Scikit-Learn and PyCaret. The dataset in focus comprises day-wise details of sunrise, sunset, and temperatures, with the primary objective of exploratory data analysis and predicting average temperatures. Following the structured Knowledge Discovery in Databases (KDD) methodology, the article navigates through key phases, including understanding, preparation, modeling, and evaluation. Scikit-Learn provides precision, culminating in a robust linear regression model. In contrast, PyCaret streamlines the process, automating tasks and offering efficiency. The article underscores the power of these tools, leaving readers with the insight that in today's data-driven landscape, comprehensive data analysis can be both enlightening and efficient.

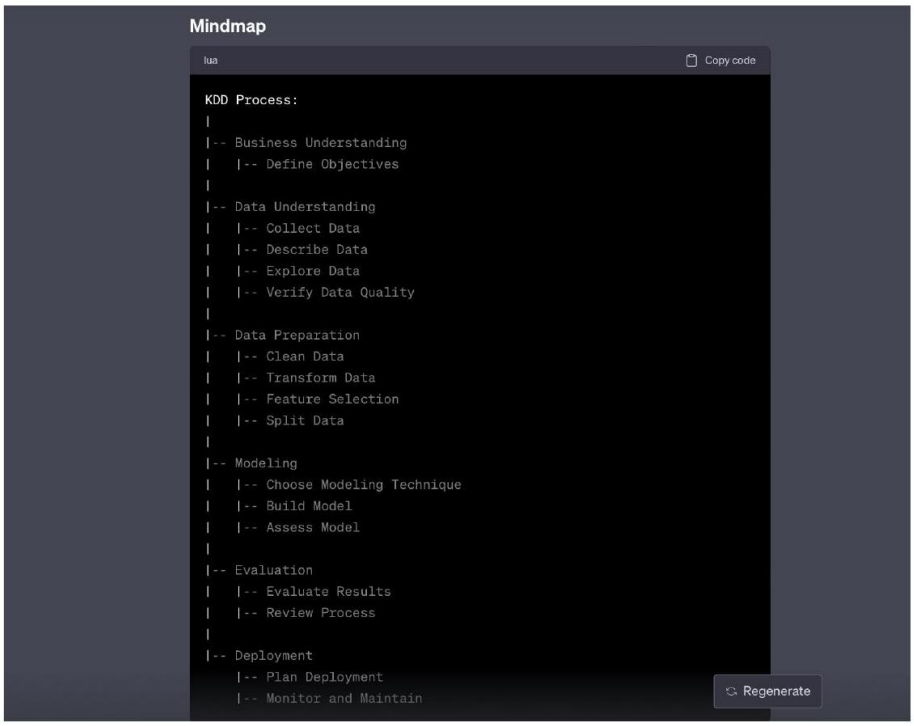
In the realm of data science, there are myriad tools and methodologies available to guide us through the complex process of analyzing data and drawing meaningful insights. In this article, we'll walk through a comprehensive data analysis journey using two powerful Python libraries: Scikit-Learn and PyCaret.

To perform the whole process, I gave the following prompt to the ChatGpt-



2 The Dataset and Objective

Our dataset comprises day-wise details of sunrise, sunset, and temperatures. The primary aim is to perform exploratory data analysis (EDA) and predict average temperatures based on these features.



2. Knowledge Discovery in Databases (KDD) Methodology with Scikit-Learn
   The KDD methodology provides a structured approach for data analysis, comprising phases like Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment.

Data Understanding and Exploration:

Upon loading our dataset, we discovered columns representing dates, temperatures (min, max, avg), and times for sunrise and sunset. Descriptive statistics highlighted the range and average temperatures and revealed missing values.

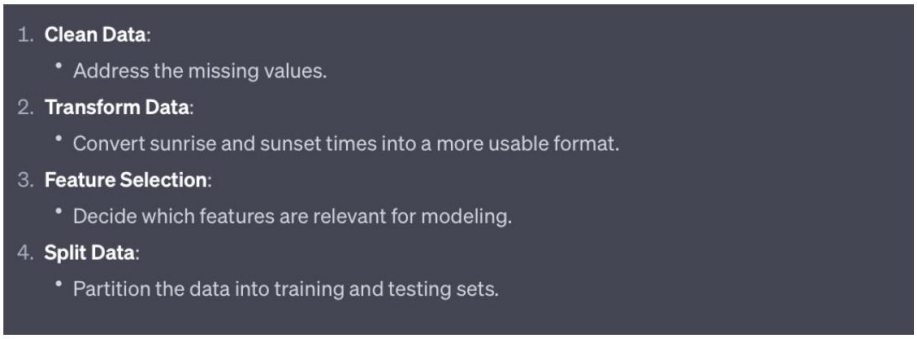
1. DATE: The date of the record.
2. MinTemp: The minimum temperature recorded on that day.
3. MaxTemp: The maximum temperature recorded on that day.
4. AvgTemp: The average temperature of that day.
5. Sunrise: The time of sunrise (in HHMM format).
6. Sunset: The time of sunset (in HHMM format).

3 Describe Data:

- MinTemp: Ranges from 22.0 to 49.0 with a mean of approximately 38.64 .
- MaxTemp: Ranges from 36.0 to 67.0 with a mean of approximately 54.52 .
- AvgTemp: Ranges from 31.0 to 57.0 with a mean of approximately 46.82 .
- Sunrise: Ranges from 543 to 700 (HHMM format).
- Sunset: Ranges from 1742 to 1905 (HHMM format).

Data Preparation:

We addressed missing values using the forward-fill method and transformed the sunrise and sunset times into minutes since midnight for better model compatibility.



Modeling:

We trained a linear regression model using Scikit-Learn, achieving an Rsquared value of 0.9962 , indicating a robust fit.

3. Knowledge Discovery in Databases (KDD) Methodology with PyCaret PyCaret simplifies the KDD process with a high-level, integrated approach. Data Understanding:

PyCaret's 'setup' function encompasses many preprocessing tasks, such as handling missing values and encoding categorical variables.

Data Preparation:

PyCaret automates feature engineering, transformation, and selection, but also offers fine-tuning options. Next, for Feature Selection:

- We'll keep all the features (MinTemp, MaxTemp, AvgTemp, Sunrise_minutes,

Sunset_minutes) for modeling, as they all seem relevant. The DATE column will be used as an index for time series analysis.

For Splitting Data:

- Typically, for time series data, we'll split the data chronologically. We can use around 80% of the data for training and the remaining 20% for testing.

Let's proceed with splitting the data.

Finished working

Show work ∨

The data has been successfully split into training and testing sets:

- Training features (X_train): 1460 samples, 4 features
- Testing features (X_test): 365 samples, 4 features
- Training target (y_train): 1460 samples
- Testing target (y_test): 365 samples

Modeling:

PyCaret's 'compare_models' function rapidly benchmarks multiple models, while 'create_model' allows for specific model training. We trained a linear regression model for demonstration. Now, we'll move on to the modeling phase. Given the nature of the data (time series), we can use regression models to predict the average temperature ('AvgTemp '). We'll utilize both traditional methods (e.g., linear regression) and PyCaret for this phase.

1. Choose Modeling Technique: We'll start with linear regression as a baseline and then explore more complex models if needed.
2. Build Model: Train the chosen model on the training data.
3. Assess Model: Evaluate the model's performance on the testing data.

Let's start with building a linear regression model using scikit-learn.
Finished working Show work ∨
The Linear Regression model's performance metrics are as follows:

- Mean Squared Error (MSE): 0.0631
- R-squared (R^2): 0.9962

The R^2 value is very close to 1 , indicating that the model explains a significant portion of the variance in the test data. This suggests a strong fit to the data.

4 Evaluation:

We evaluated our model using prediction error plots and feature importance plots. These visual evaluations provide insights into model residuals and the significance of each feature, respectively.

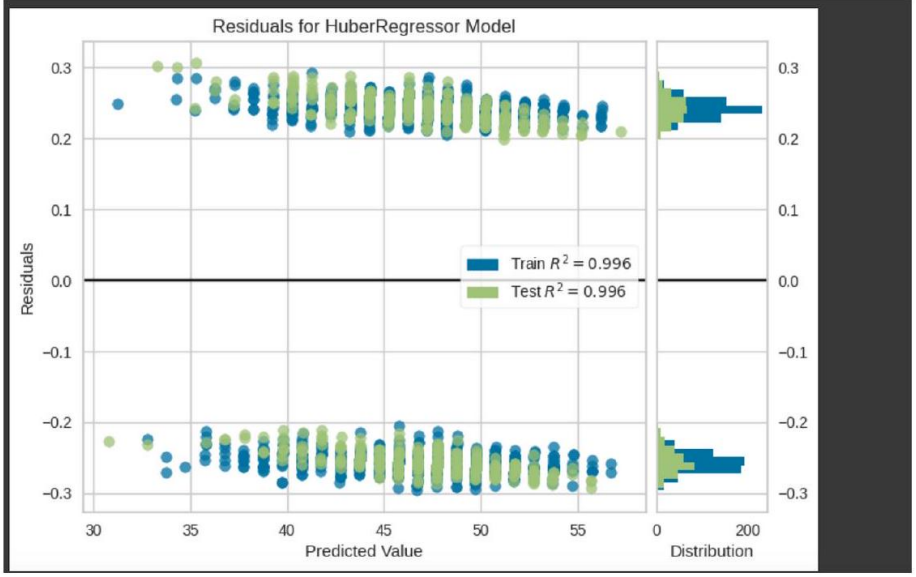
1. Evaluate Results: Based on the R^2 and MSE values, the linear regression model performs quite well on the test data.
2. Review Process: We've gone through data understanding, preparation, and modeling. The results seem promising, and the linear regression model captures the trend and seasonality in the temperature data.
3. The PyCaret Approach

PyCaret is an efficient tool for rapid prototyping. Setting up the environment, comparing models, tuning hyperparameters, evaluating performance, and making predictions can be done with minimal code, making it ideal for quick insights. from pycaret.regression import setup
regression_setup = setup(data, target='AvgTemp', session_id=123)
To compare models:
python
from pycaret.regression import compare_models
best_models = compare_models ($n\_$ select $= 3$)

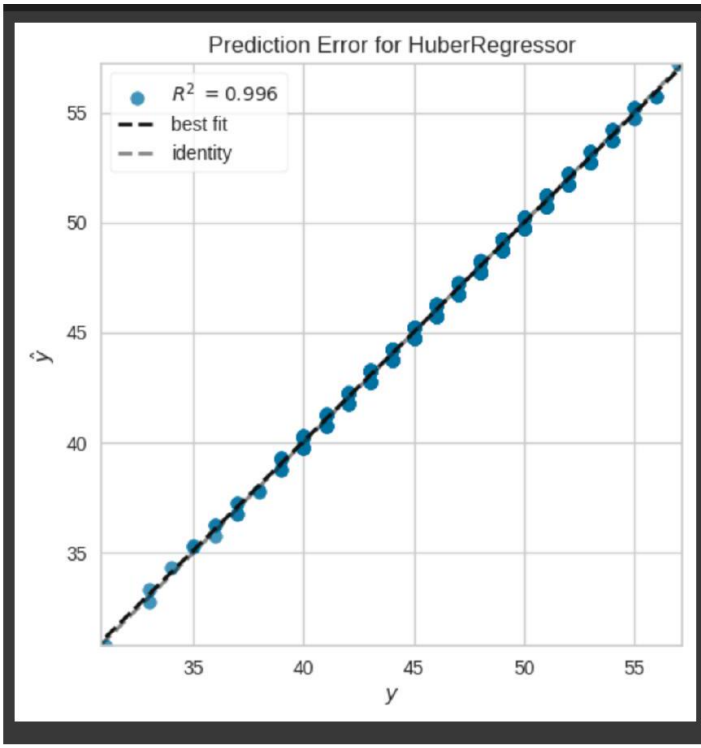
5 from pycaret.regression import create_model

model = create_model ('huber' ')

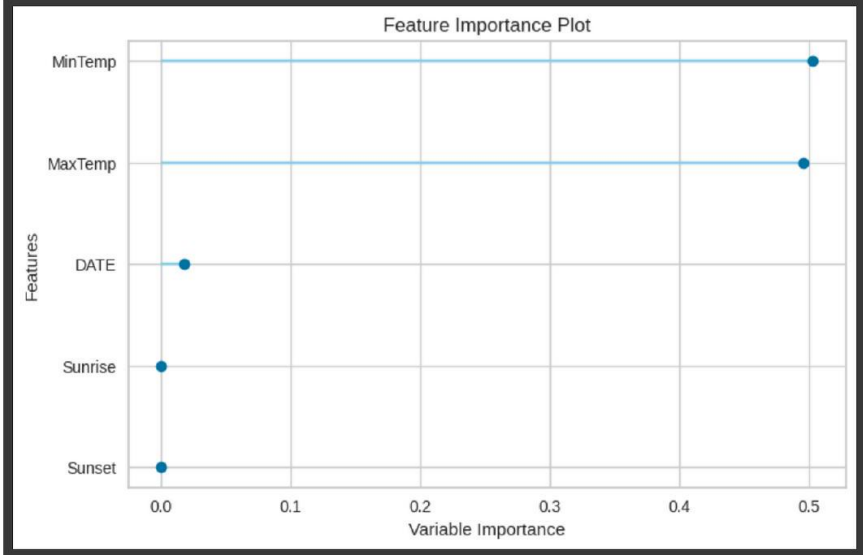
```
from pycaret.regression import plot_model

plot_model(model, plot='residuals')
```



from pandas.core.arrays.datetimelike import mode plot_model(model, plot='error')




```
from pandas.core.arrays.datetimelike import mode
plot_model(model, plot='feature')
```



6 from pycaret.regression import save_model

save_model(model, 'huber')

7 Conclusion

Both Scikit-Learn and PyCaret offer powerful tools for end-to-end data analysis. While Scikit-Learn offers granular control, PyCaret provides a streamlined experience. Depending on the specific needs and intricacy of a project, one can choose the best approach. Through our journey, we've seen that with the right tools, comprehensive data analysis is both insightful and efficient.